Česká zemědělská univerzita v Praze
Fakulta technická

Model regulace tepelné soustavy

Diplomová práce

Vedoucí diplomové práce: Ing. Künzel Gunnar
Autor: Miroslav Brabec

PRAHA 2014

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra elektrotechniky a automatizace

Technická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Brabec Miroslav

Informační a řídící technika v agropotravinářském komplexu

Název práce

Model regulace tepelné soustavy

Anglický název

Model cotrol of termal object

Cíle práce

Cílem práce je formulovat model vyhřívaného prostoru pece a její regulace v prostředí MATLAB-Simulink a tento model ověřit na reálné soustavě.

Metodika

Seznámit se s dostupným objektem možnostmi jeho regulace
Stanovit technické požadavky na průběh teploty
Zvolit vhodné typy regulátorů
Formulovat a řešit matematický model regulace teploty na PC
Provést měření na reálné soustavě a zhodnotit shodu chování modelu a objektu

Osnova práce

Objekt regulace
Požadavky na regulaci teploty
Vhodné regulátory
Model celého regulačního obvodu
Měření v regulačním obvodu
Porovnání chování modelu a objektu regulace

Rozsah textové části

50 str. včetně příloh

Klíčová slova

Tepelná soustava, pec, model, regulace, regulátor, , MATLAB- Simulink

Doporučené zdroje informací

Valter, J.: Regulace v praxi, BEN,Praha,2010
Schmid, D. a kol.:Řízení a regulace pro strojírenství a mechatroniku, Sobotáles, Praha, 2006
Roubal, J.: Regulační technika v příkladech, BEN,Praha,2011
časopisy Automatizace, Automa.

Vedoucí práce

Künzel Gunnar, Ing.

Termín zadání

listopad 2012

Termín odevzdání

duben 2014

prof. Ing. Jaromír Volf, DrSc.
Vedoucí katedry

prof. Ing. Vladimír Jurča, CSc.
Děkan fakulty

V Praze dne 9.2.2013

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literární prameny a publikace) uvedené v přiloženém seznamu.

V Praze dne _____ _____

podpis

Poděkování

Rád bych poděkoval vedoucímu diplomové práce Ing. Gunnaru Künzelovi za konzultace s textovou částí. Poděkování patří také Ing. Miloslavu Lindovi, Ph.D. za zapůjčení přístrojů potřebných k realizaci praktické části mé práce, pomoc s měřením a konzultace při psaní diplomové práce. Zvláštní poděkování patří RNDr. Pavlu Patákovi za korekci textu a pomoc se sazbou v systému LaTeX.

Abstrakt a klíčová slova

Abstrakt:
Práce se zabývá návrhem a realizací regulátoru pro tepelnou soustavu, konkrétně zlatnickou pec. Teoretická část popisuje, jak a čím se teplota měří a základy a možnosti její regulace v soustavě. Na základě změřených dat ze zlatnické pece byl vytvořen návrh regulátoru a model soustavy. Tento návrh byl odsimulován v programu Matlab před tím, než došlo k samotnému vyhotovení regulátoru. Součástí finální realizace regulátoru je i program pro platformu Netduino. Podařilo se sestrojit funkční regulátor, pomocí kterého bylo při počáteční teplotě 200°C dosaženo ustálení na požadované hodnotě 400°C po méně než třech hodinách.

Klíčová slova: Tepelná soustava, pec, model, regulace, regulátor, Matlab - Simulink.

Model control of thermal object

Abstract and key words

Abstract: The main goal of the thesis is a creation and implementation of a thermal system controller, the thermal system in question being goldsmith furnace. Theoretical part describes temperature measurements and basic options for thermal control in thermal systems. Series of measurements served as a basis for creating a model of the furnace and for a suggestion of a thermal controller. First the function of the suggested controller was simulated in the computer software Matlab-Simulink. After necessary adjustments a real controller was fabricated. The part of the final realization is also a control software for platform Netduino. As the final test has shown, starting on 200°C the controller was able to achieve stable target temperature of 400°C after less than three hours.
Key words: Termal object, furnace, model, control, controller, Matlab - Simulink.

Obsah

Úvod **1**

1. Cíle a metodika práce **2**
1.1. Cíle práce . . . 2
1.2. Metodika . . . 2

2. Teoretická část **4**
2.1. Měření teploty . . . 4
2.1.1. Teplota . . . 5
2.1.2. Snímač teploty . . . 5
2.1.3. Termoelektrické články . . . 6
2.1.4. Materiály pro zhotovení termoelektrických článků . . . 7
2.1.5. Termoelektrické snímače pro nízké teploty . . . 9
2.1.6. Termoelektrické snímače pro vysoké teploty . . . 10
2.2. Regulace . . . 10
2.2.1. Regulační člen . . . 11
2.2.2. Akční člen . . . 12
2.2.3. Pulzně šířková modulace . . . 12
2.2.3.1. Pulzní regulátory . . . 13
2.2.3.2. Krokové regulátory . . . 13
2.2.3.3. Analogové (spojité) regulátory . . . 14
2.2.3.4. Číslicové regulátory . . . 15
2.2.3.5. Číslicová regulace . . . 16
2.2.4. Aproximace . . . 17
2.2.4.1. Avramiho rovnice . . . 17
2.2.4.2. Strejcova metoda . . . 18
2.2.5. Matlab - Simulink . . . 20
2.2.6. Matlab - Pidtool . . . 20

3. Použité přístroje **22**
3.1. Použité přístroje pro měření . . . 22

3.2. Použité prvky při realizaci regulátoru 26
3.2.1. Průběh měření 29

4. Aproximace a návrh regulátoru **34**
4.1. Aproximace naměřených křivek a výpočet S křivek 34
4.2. Naměřené hodnoty 37
4.3. Matlab - Simulink 37
4.3.1. Naměřené hodnoty 39

5. Realizace regulátoru **43**
5.0.1. Programování Netduina 44
5.0.2. Aplikace pro PC 49

6. Porovnání modelu z Matlabu a regulované soustavy **51**
6.1. Ověření funkčnosti regulátoru na tepelné soustavě 51
6.2. Porovnání výsledků 53

7. Závěr **55**

Seznam zkratek **59**

Seznam obrázků **60**

Seznam tabulek **62**

A. Kód pro netduino **I**

B. Kód aplikace pro počítač **VIII**

C. Obsah přiloženého CD **XV**

Úvod

Cílem práce je vytvořit regulátor tepelné soustavy, v tomto případě zlatnické pece. Zvolen byl dále popsaný postup. Na jiné peci s regulátorem byla nejprve naměřena data, která slouží hlavně jako ukázka správné regulace. Následně byla změřena charakteristika ohřevu cílové zlatnické pece. Na základě naměřených dat byl navržen regulátor, jehož činnost byla nejprve ověřena simulací v programu Matlab. S pomocí simulace byl návrh regulátoru mírně upraven a následně realizován. Správná činnost regulátoru poté byla ověřena na zlatnické peci.

První kapitola seznamuje s použitou metodikou a cíli práce. Druhá kapitola se věnuje základům měření teploty, typům regulátorů a způsobům regulace tepelných soustav. Tato kapitola vysvětluje principy práce snímačů teploty a materiály, ze kterých se běžně vyrábí. Dále jsou popsány regulační obvody a typy regulátorů. Také je uveden postup získání přenosu z naměřené charakteristiky ohřevu pece. Na konci kapitoly se nachází základní popis použitého softwaru Matlab a jeho modulu Matlab - Simulink.

Seznam přístrojů použitých při sestavení regulátoru je uveden v třetí kapitole. V první části kapitoly jsou popsány pece a použité měřící přístroje. Následuje seznam a popis pomůcek použitých při samotné tvorbě regulátoru. Poslední část se věnuje průběhu měření na pecích, včetně schématu zapojení při měření.

Hodnoty naměřené na peci s regulátorem jsou prezentovány v čtvrté kapitole. Tyto výsledky slouží pro porovnání naměřené pece s regulátorem a zlatnické pece s realizovaným regulátorem. Další část kapitoly prezentuje výsledky měření, jejich aproximaci a zpracování v programu Matlab. Konec kapitoly se věnuje modelování systému s regulátorem v programu Matlab - Simulink.

Předposlední kapitola se týká hlavně zapojení řídícího obvodu. Také obsahuje detailní popis vybraných částí programového kódu. Celý programový kód je zevrubně popsán v přílohách.

Poslední kapitola porovnává navržený regulátor s modelem z Matlabu. Ukazuje se, že navržený regulátor plní svou funkci nejen v rámci simulace, ale i v ostrém nasazení.

1. Cíle a metodika práce

Cíle a postup jejich vypracování, kterými se bude tato práce zabývat.

1.1. Cíle práce

Cílem práce je sestavení návrhu regulátoru v programu Matlab - Simulink a jeho realizace. Podle navrženého modelu z Matlabu se sestaví regulátor tepelné soustavy, který se ověří na reálné tepelné soustavě. Tepelná soustava je v tomto případě zlatnická pec. Tato pec je bez regulátoru. Cílem je sestavit regulátor, do kterého se zadá požadovaná hodnota teploty a na tuto hodnotu se bude pec regulovat.

Regulátor bude realizován pomocí platformy Netduino Plus 2, které bude řídit proces regulace.

Výsledek navržené regulace se porovná s navrženou křivkou ze simulace.

1.2. Metodika

Postup práce při navrhování a realizaci regulátoru tepelné soustavy.

Nejprve se naměří charakteristika ohřevu na jiné peci, která slouží jako ukázka toho, jak má regulace u tepelné soustavy vypadat.

Na peci, pro kterou se má navrhnout regulátor, se naměří přechodová charakteristika, ze které se získá přenos soustavy. Získaný přenos se použije při sestavení modelu soustavy v Matlab - Simulink. Ze simulace v Matlabu se získají konstanty regulátoru potřebné pro fyzickou realizaci regulátoru.

Při realizaci regulátoru na Netduinu Plus 2 se implementují získané konstanty ze simulace do jeho programu. Programování je možné buď v jazyce Visual Basic nebo C#. Po naprogramování Netduina se vytvoří aplikace v počítači, která bude sloužit k zapnutí

a vypnutí procesu regulace. Tato aplikace bude zobrazovat aktuální naměřenou hodnotu a bude naměřené hodnoty ukládat do tabulky. Tuto tabulku s naměřenými hodnotami bude možné exportovat do formátu excel.

Funkčnost vytvořeného regulátoru se ověří na tepelné soustavě. Následně se porovná naměřená charakteristika vytvořeného regulátoru s charakteristikou z Matlab - Simulinku.

2. Teoretická část

2.1. Měření teploty

Teplota je jedna z nejdůležitějších termodynamických vlastností, které určují stav hmoty a objevuje se v mnoha fyzikálních zákonech. Existuje jen velmi málo vlastností látek, které by nebyly teplotně závislé. Ve své podstatě je teplota mírou kinetické energie pohybu molekul a atomů, přičemž molekuly na sebe navzájem narážejí a rychlost jejich pohybu se stále mění v čase. V pevném skupenství molekuly neuspořádaně kmitají kolem rovnovážné polohy, v tekutém skupenství je pohyb molekul neuspořádaný v celém objemu. Pokud na hmotu nepůsobí okolní prostředí, průměrná rychlost pohybu molekul je konstantní. Tato průměrná rychlost je závislá na teplotě a termodynamická teplota libovolného tělesa je přímo úměrná kinetické energii molekul neboli přímo úměrná jejich hmotnosti a kvadrátu rychlosti pohybu. Za nejnižší teplotu se pokládá absolutní teplotní nula neboli nula termodynamické stupnice (0 K), během níž veškerý pohyb ustává. Pro stanovení teploty se používají teplotní závislosti jiných fyzikálních veličin. Volí se takové fyzikální jevy, u nichž lze závislost veličiny matematicky vyjádřit teplotní stupnicí. [1, 8]

Znalost teploty je nutná v mnoha oblastech lidské činnosti. Je základem pro určení bezpečnosti a spolehlivosti v energetice. Využití má v leteckém, automobilovém i železničním průmyslu, ale i v lékařství a mnoha dalších vědních disciplínách. Slouží například ke měření teploty ložisek u vlaků, v metalurgii pro výrobu kvalitních odlitků.

Každý měřící přístroj nebo měřící systém pro měření libovolné fyzikální veličiny má stanovený rozsah teploty, v němž je zajištěna mezní hodnota nejistoty přístroje nebo je stanovena změna údaje měřené veličiny vyvolaná teplotou. Teplota je jedním ze zdrojů nejistoty ovlivňující nejistotu měření jiných veličin. [8]

2.1.1. Teplota

Teplota je stavová veličina určující stav termodynamické rovnováhy, tj. stav, kdy v soustavě těles izolované od okolního prostředí neprobíhají žádné makroskopické změny a všechny fyzikální veličiny, jimiž je stav soustavy popsán, nezávisejí na čase. Stav termodynamické rovnováhy bývá charakterizován termodynamickou teplotou, která musí být stejná pro všechny části izolované soustavy. Fyzikální veličiny teplo a teplota jsou dvě rozdílné veličiny a neměly by se zaměňovat. Teplo je forma energie související s pohybem částic dané soustavy těles, nejde o stavovou veličinu, protože nezávisí jen na přítomném stavu soustavy, ale je potřeba znát celou její minulost. Měření teploty je měření nepřímé, protože teplota se nedá měřit přímo, ale pouze prostřednictvím jiných veličin. [1]

2.1.2. Snímač teploty

Snímač teploty je funkční prvek tvořící vstupní blok měřícího řetězce, který je v přímém styku s měřeným prostředím. Snímač teploty je samostatná konstrukční část teploměrového zařízení s teplotním čidlem. Čidlo teploty je část, která převádí teplotu na jinou vhodnou fyzikální veličinu. Jako snímač teploty se označuje detektor tepelného záření nebo teploměr. [1, 5]

Podle [1]dělíme snímače dle fyzikálních principů na:

odporové
termoelektrické
polovodičové
dilatační
optické
radiační
chemické
šumové
akustické
magnetické
a na další jako jsou například kapacitní, aerodynamické.

Podle styku s měřeným prostředím se snímače dělí na dotykové a bezdotykové. Dle transformace signálu se snímače teploty dělí na aktivní, které se působením teploty chovají jako zdroj elektrické energie (termoelektrické články), a na pasivní, u kterých je s výjimkou chemických indikátorů teplot nutné elektrické napájení pro transformaci teploty na jinou fyzikální veličinu, neboť měření teploty je vždy měřením nepřímým. [8]

Statické vlastnosti snímačů teploty dle [1]:

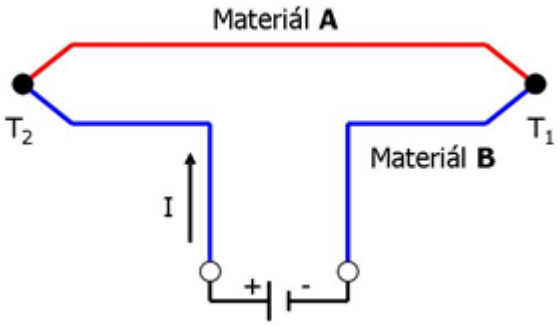
- Statická charakteristika snímače - dána funkční závislostí mezi měřenou veličinou a transformovanou veličinou v časově ustáleném stavu.
- Práh citlivosti snímače - dán hodnotou měřené veličiny, při které je na výstupu snímače signál odpovídající střední kvadratické odchylce šumu snímače.
- Dynamický rozsah snímače teploty - je dán intervalem přípustných hodnot snímané veličiny, ohraničený prahem citlivosti a maximální hodnotou měřené veličiny.
- Reprodukovatelnost snímače - dána odchylkou naměřených hodnot při krátkodobě neměnné měřené veličině a neměnných rušivých vlivech okolí.
- Rozlišitelnost snímače - je poměr věrohodné měřené hodnoty a prahu citlivost.

Dynamické vlastnosti snímačů teploty dle [1]:
Dynamické vlastnosti snímačů teploty je nutné znát pro analýzu a syntézu měřících a regulačních systémů. V regulátorech teploty nebo v systémech monitorování mezních stavů musí být použity takové snímače teploty, aby výstupní signál sledoval s minimálním zkreslením teplotu.

Dynamické vlastnosti snímačů lze popsat rovnicemi prvního, druhého a výjimečně vyššího řádu. Na grafické zobrazení dynamických vlastností snímačů se používá přechodová charakteristika (odezva na jednotkový skok) nebo rychlostní charakteristika (odezva na změnu teploty konstantní rychlostí).

2.1.3. Termoelektrické články

Termoelektrické články jsou založeny na Seebeckově jevu, převodu tepelné energie na elektrickou. Seebeckův jev (obr.(2.1)) se projevuje vznikem termoelektrického napětí ve spojených vodičích z rozdílných materiálů, které mají rozdílnou teplotu (T1, T2). Z teplejší části obou vodičů difundují ohřáté částice do studené části, pokud je ohřátých částic na studeném konci menší hustota. To samé platí i opačně.



Obr. 2.1.: Schéma Seebeckova jevu. [12]

Termoelektrický článek je složen vždy ze dvou vodičů (polovodičů). V uzavřeném obvodu realizovaném ze dvou materiálově různých vodičů nebo polovodičů protéká elektrický proud tehdy, pokud oba spoje mají rozdílnou teplotu. Pokud obvod rozpojíme na kterémkoliv místě, bude na vzniklých svorkách elektrické napětí. [1]
Termoelektrické články se využívají pro měření teplot nebo jako generátor elektrické energie a jako čidlo spojitých regulátorů teploty. [2, 5]

2.1.4. Materiály pro zhotovení termoelektrických článků

Při výběru materiálu pro termoelektrický snímač je třeba splnit některé základní požadavky, ke kterým především patří dle [2]:

- Závislost termoelektrického napětí na teplotě by se měla blížit lineárnímu průběhu.
- Použitý materiál musí být odolný proti chemickým, mechanickým a korozním vlivům.
- Výstupní termoelektrické napětí by mělo být dostatečně velké, neboť s klesajícím napětím klesá přesnost.

Materiál se volí podle požadované přesnosti a rozsahu teplot při měření. Důležitá je také časová stálost, popř. střední doba životnosti snímače. Neproměnná stálost snímače je podmínka, která se za vysokých teplot těžko dodržuje. Dochází k rekrystalizaci v místě spoje, případně k jeho stárnutí. Tato problematika se vyskytuje především u měření a regulace v pecích a pících. [2]

Vlastnosti a použitelnost termoelektrických článků podle [1]:

- Typ K: má složení NiCr-NiAl (chromel-alumel) a je vhodný pro oxidační a inertní atmosféru, je necitlivý pro neutronový tok, není vhodný pro měření ve vakuu. [1]
- Typ T: má složení Cu-CuNi (měď-konstantan) a je nejlepší termoelektrický článek pro nízké teploty kryogenní aplikace, v redukční, oxidační atmosféře a ve vakuu ho lze použít až do teploty 700°C. [1]
- Typ J: má složení Fe-CuNi (železo-konstantan) a je vhodný pro oxidační, redukční i inertní atmosféru a vakuum. V redukční atmosféře ho lze použít bez ochranného krytí. [1]
- Typ N: má složení NiCeSi-NiSiMg (nicrosil-nisil), má velmi stabilní charakteristiku až do 1300°C a je vhodný pro cyklické změny teploty, dále je vhodný pro jadernou energetiku, neboť je odolný vůči neutronovému toku. [1]
- Typ E: má složení NiCr-CuNi (chromel-konstantan), má nevyšší hodnotu termoelektrického koeficientu, dále je vhodný pro vakuum a středně oxidační atmosféru, kde se dá využít bez ochranného krytí. [1]
- Typ R: má složení PtRh13-Pt a je používán pro měření vysokých teplot až do 1780°C, je odolný vůči oxidaci a korozi, ale vždy musí být v ochranném provedení. [1]
- Typ S: má složení PrRh10-Pt, jinak má charakteristiky stejné jako u typ R. [1]
- Typ B: má složení PtRh30-PtRh6 a je použitelný až od teploty 100°C (v rozsahu do 300°C má velmi malou citlivost), má obdobné vlastnosti jako typ R a S, ale při teplotách nad 1200°C je stabilnější. [1]
- Typ G: má složení W-WRh a je vhodný pro extrémně vysoké teploty, obvykle se nepoužívá pro rozsahy teplot pod 400°C, je chemicky stabilní a vhodný k použití jak v oxidační tak v inertní atmosféře, ve vakuu a ve vodíku. [1]
- Typ C: má složení WRh5-WRh26 a má obdobné vlastnosti jako typ G.

Jednotlivé typy termočlánků označené písmenem mají svůj teplotní rozsah. V následující tabulce (tab.(2.1)) jsou uvedeny některé z nich.

Tabulka 2.1.: Rozsahy měřených teplot u vybraných termočlánků.

Typ	Rozsah (°C)	
	Od	Do
K	-200	1250
T	-200	350
J	0	750
N	-270	1300
E	-200	900
R	0	1450
S	0	1450
B	0	1700
G	0	2320
D	0	2300
C	0	2320

2.1.5. Termoelektrické snímače pro nízké teploty

Pro měření nízkých teplot -250 až 500°C se používá běžně termoelektrický snímač Cu-Ni. Při přechodu přes 0°C se mění znaménko termoelektrického napětí snímače. Pro vyšší teploty (v rozsahu nízkých teplot) je vhodná dvojice Fe-CuNi. Konstantan má podle normy 45 % Ni, 55 % Cu a nepatrný obsah jiných přísad (Mn, Si, Co, Mg), jehož zvětšení může nepříznivě ovlivnit termoelektrické vlastnosti snímače. Tyto přísady jsou vhodné hlavně pro zvětšení odolnosti proti korozi. Velmi choulostivá je konstantanová větev v sirném prostředí. Někdy bývají jako konstantan označovány též slitiny s poněkud odlišným složením, které se pro realizaci snímačů tohoto typu nehodí. [2, 7]

Měď používaná na výrobu termoelektrických článků má být elektrolyticky čistá. Na čistotě mědi velmi záleží, protože jakékoliv příměsi ovlivňují velmi podstatně její termoelektrické vlastnosti. Poměrně úzký teplotní rozsah je omezen malou odolností mědi, která snadno oxiduje. [2]

Znečištění železa (S, Si, Mn a zvláště C) smí být pouze několik setin procenta. Železo je náchylné ke korozi zvláště v prostředích plynných spalin. Intenzita koroze závisí jednak na kvalitativním složení spalin, dále pak na jejich teplotě. Pro různé prostředí, při určitých teplotách a stanovíme-li si přípustnou toleranci termoelektrického napětí, můžeme určit dobu života snímače. [11]

2.1.6. Termoelektrické snímače pro vysoké teploty

Pro měření vyšších teplot se používají termoelektrické články NiCr-Ni. Niklová část má 95 % Ni, zbytek je tvořen dezoxidačními a jinými přísadami. Část niklochromová má mít 85 % Ni, 10 % Cr a zbytek tvoří dezoxidační a jiné přísady. Je velmi těžké zachovat předem stanovené složení, obzvláště u obsahu chrómu. Odchylky ve vlastnostech termoelektrických snímačů jsou způsobeny odchylkami v jejich složení, materiálu výroby, různých taveb nebo různými výrobci. [2]

Chyba u termoelektrického článku způsobená teplotou srovnávacích (studených) konců

Vliv teploty srovnávacích konců termoelektrických článků podle [2] je možno omezit dvěma způsoby:

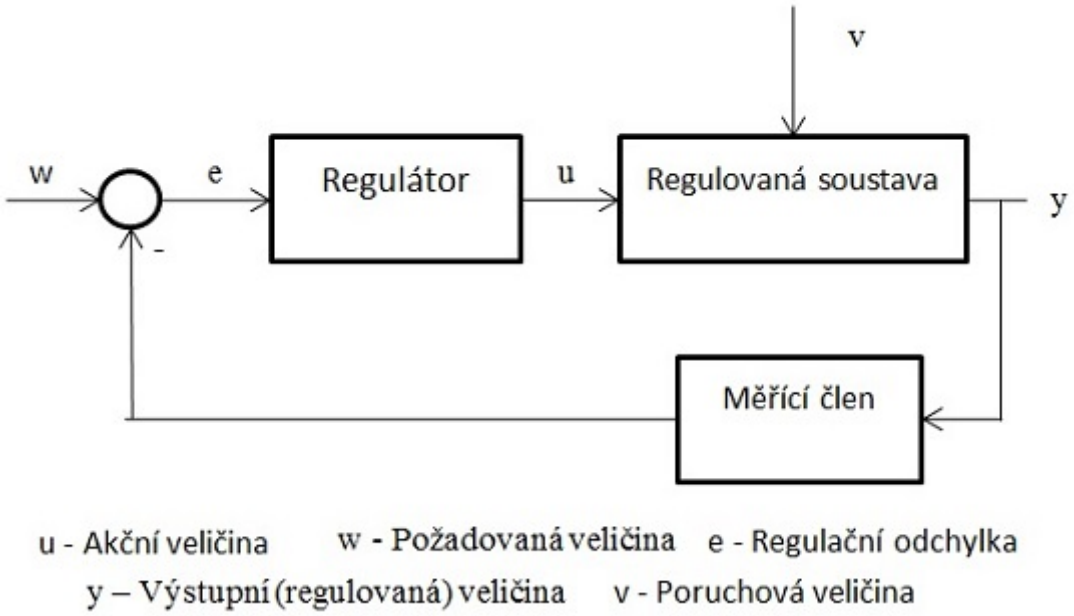
- konstantní teplotou svorkovnice,
- kompenzací pomocí dalšího termočlánku umístěného na studeném konci.

Spojení mezi vlastním termočlánkem a místem teploty studených spojů se realizuje pomocí kompenzačního (prodlužovacího) vedení. U obecných kovů termoelektrického článku se používá kompenzační vedení ze stejných kovů. Pro články z drahých kovů je kompenzační vedení tvořeno z náhradních kovů, používá se měď a nikl. Podmínkou u kompenzačního vedení je, aby termoelektrické napětí tohoto vedení bylo shodné nebo blízké s termoelektrickým napětím vlastního termoelektrického článku. [2, 11]

2.2. Regulace

Regulace je proces, který udržuje regulovanou veličinu na požadované hodnotě. Regulace probíhá v uzavřené smyčce (obr.2.2) tvořené zpětnou vazbou regulované veličiny. Poruchová veličina způsobuje v regulační smyčce nestabilitu, která je příčinou odchýlení skutečné hodnoty od požadované hodnoty o tzv. regulační odchylku. V regulační smyčce je regulovaná veličina snímána měřícím členem a srovnávána v komparátoru s požadovanou hodnotou. Rozdíl požadované hodnoty a skutečné hodnoty je regulační odchylka, která je regulačním členem transformována na akční veličinu a ta je výstupní veličinou regulátoru a zároveň vstupní veličinou do regulovaného systému ovládaného pouze pomocí akčního členu. [7, 11]

Ve stabilním stavu regulační smyčky je regulační odchylka velmi malá nebo nulová. Při poruchách, nebo změnách nastavení požadované hodnoty naroste regulační odchylka, která je pak změnou požadované veličiny minimalizována. [7]



Obr. 2.2.: Schéma regulačního obvodu.

2.2.1. Regulační člen

Vlastnosti regulačních členů je možné popisovat v ustáleném stavu, kdy se jedná o statické vlastnosti, nebo pokud se mění vstupní a výstupní veličina, pak jde o dynamické vlastnosti regulačních členů nebo systémů. [9]

Statické vlastnosti regulačních členů se nejčastěji vyjadřují statickou charakteristikou, závislost mezi vstupní veličinou a výstupní veličinou v ustáleném stavu.

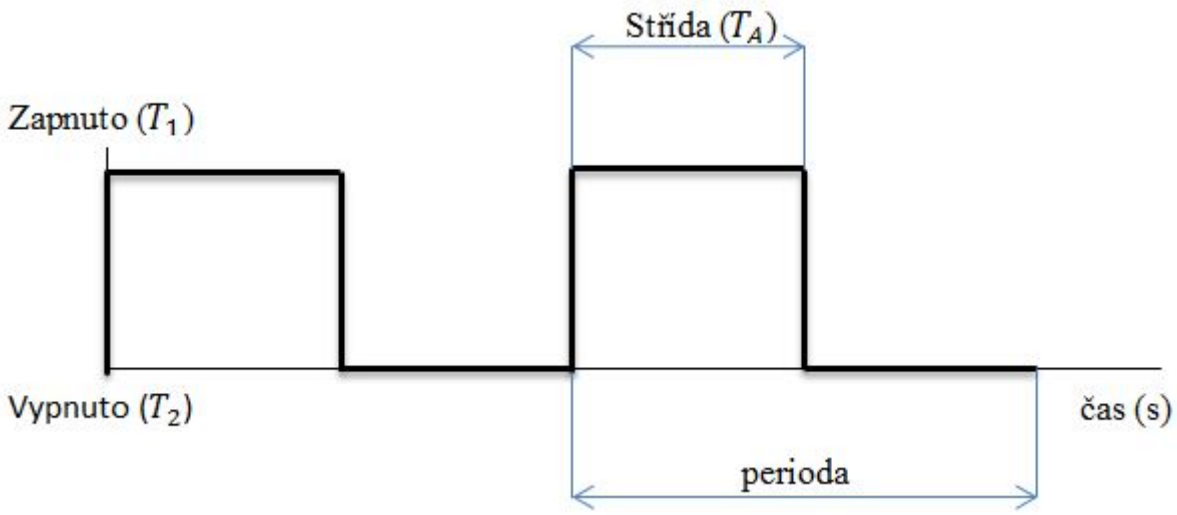
Při snímání statické charakteristiky musí být dosáhnuto ustáleného stavu u vstupní i výstupní veličiny. Ustálení znamená, že musí proběhnout přechodový děj, teprve poté se odečte příslušná hodnota vstupní nebo výstupní veličiny. [7, 9]

2.2.2. Akční člen

Skládá se z pohonu a regulačního orgánu. Regulační orgán slouží k nastavení protékané energie regulovanou soustavou, může to být například šoupátko, stavidlo nebo ventil. Pohon dodává energii regulačnímu orgánu, který pak může měnit své nastavení, například polohu otevření šoupátka nebo otevřít ventil.

2.2.3. Pulzně šířková modulace

Pulzně šířková modulace (Pulse Width Modulation), dále jen PWM, je regulace, u které přenosový signál nese informaci o přenášeném stavu zapnuto nebo vypnuto. Poměru mezi těmito stavy během jedné periody se říká střída. Omezením pro PWM je to, že přenos informace je vždy omezen na relativní vyjádření 0 – 100 %, to znamená, že musí být znám poměr mezi skutečnou hodnotou a procentuálním vyjádřením. [21]



Obr. 2.3.: Demonstrace pojmů u PWM při nastavení střídy na 50%.

Na obr.2.3 je vidět jak vypadá PWM při nastavení 50 % periody. Střída je v hodnotě zapnuto polovinu času z délky periody. V případě PWM 100 %, by byla hodnota střídy zapnuto po celou dobu periody. To znamená, že čím menší hodnota nastavení střídy, tím méně bude zařízení regulované pomocí PWM cyklu ve stavu zapnutém.

Výpočet výkonu střídy:

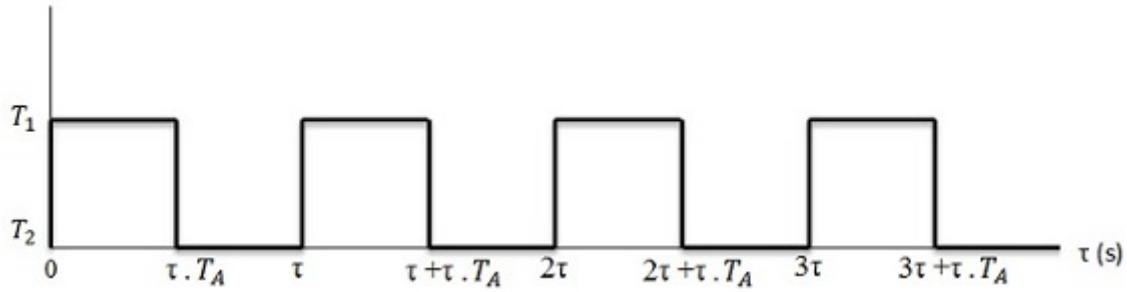
$$P = U \cdot I \tag{2.1}$$

P (W) - výkon
T_1 (-) - zapnuto
T_2 (-) - vypnuto
Hodnota středního výkonu se dá popsat i pomocí integrálu, ale vzhledem k jednoduchému průběhu ve tvaru obdélníka postačí následující vzorec.

Střední hodnota výkonu:

$$P_s = \frac{P_1 \cdot T_A}{\tau} \tag{2.2}$$

P_s (W) - střední výkon
τ (s) - perioda
T_A (s) - aktivní část
P_1 (W) - výkon aktivní části



Obr. 2.4.: Chování PWM.

Na obr.2.4 je vidět, jak vypadá PWM cyklus při nastavení střídy na 50 % v τ, 2τ, 3τ.

2.2.3.1. Pulzní regulátory

Mění řídící veličinu přepínáním dvou nebo několika stavů. Dvoustavové nebo třístavové regulátory se například používají k regulaci teploty. Dvoustavové mají dva rozlišitelné stavy a třístavové mají tři rozlišitelné stavy. [9, 11]

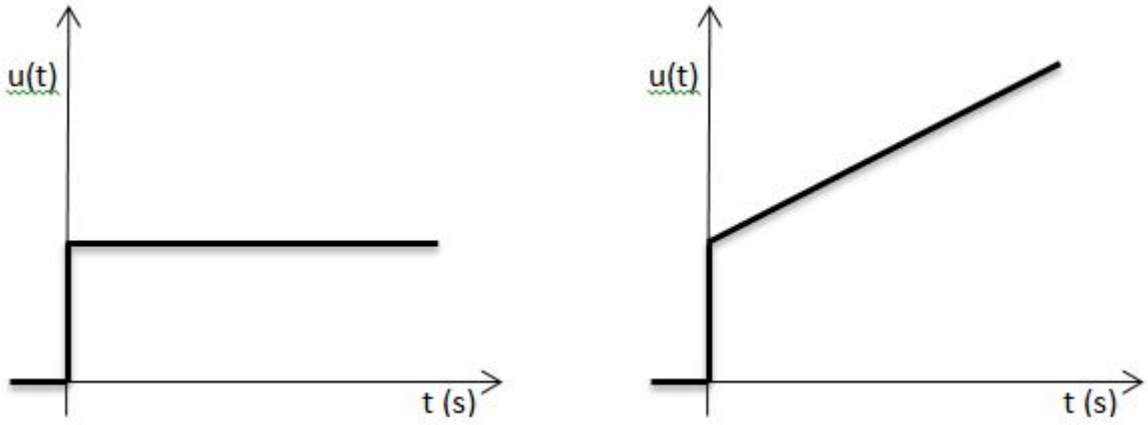
2.2.3.2. Krokové regulátory

Jde o přepínací regulátor, který nastavuje akční člen po krocích. Může být tvořen třístavovým regulátorem nebo servomotorem. [11]

2.2.3.3. Analogové (spojité) regulátory

Mají plynule nastavovaný akční člen. Jejich základním prvkem je většinou operační zesilovač. Jde o regulátory, které mohou natavit řídící veličinu na kteroukoliv hodnotu mezi oběma krajními hodnotami spojitého rozsahu. [11]

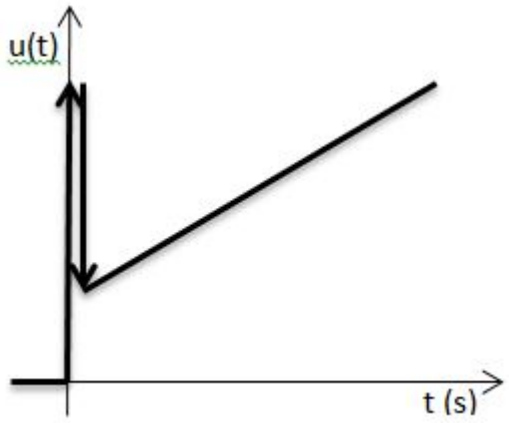
Nejdůležitějšími analogovými regulátory jsou P, PI a PID. P regulátor (proporciální regulátor), na obr.2.5 vlevo, má lineární závislost regulované veličiny na nastavovací veličině a pracuje bez zpoždění a s trvalou regulační odchylkou. U PI regulátoru, na obr.2.5 vpravo, v počátku regulačního pochodu převládá vliv proporcionální složky, s narůstajícím časem převládá vliv integrační složky. V uzavřeném regulačním obvodu odstraňuje trvalou regulační odchylku. [8, 9, 11]



Obr. 2.5.: Přechodová charakteristika P a PI regulátoru.

PID regulátor (obr.2.6) je tvořen váženým součtem proporciální odezvy na regulační odchylku P, I a D složky. Při návrhu regulátoru je potřeba nastavit tři parametry, nastavení proporciální, integrační a derivační složky. Při tom je třeba dodržet kritéria kvality regulace, která jsou rychlost regulace, přesnost, překmit. Regulační obvod musí být vždy stabilní. [8, 11]

V uzavřeném regulačním obvodu odstraňuje vlivem I složky trvalou regulační odchylku a vlivem D složky zlepšuje stabilitu regulačního obvodu. V počátku přechodového děje převládá derivační složka regulátoru, s narůstajícím časem převládá integrační složka regulátoru. [8]



Obr. 2.6.: Přechodová charakteristika PID regulátoru.

Rovnice přenosu PID regulátoru dle [9]:

$$G(s) = r_0 + \frac{r_{-1}}{s} + r_1 s = r_0 \left(1 + \frac{1}{\frac{r_0}{r_{-1}}} + \frac{r_1}{r_0} s \right) = r_0 \left(1 + \frac{1}{T_i s} + T_d s \right) \tag{2.3}$$

Význam konstant dle [9]:
r_0 (-) - proporciální
r_{-1} (-) - integrační
r_1 (-) - derivační

Kde:
r_0 (-) je zesílení regulátoru
T_i (s) je integrační konstanta regulátoru
T_d (s) derivační časová konstanta regulátoru

2.2.3.4. Číslicové regulátory

Číslicové regulátory bývají řízené mikropočítačem, jejich chování je určeno programem v jejich paměti. Zpracovávají číslicový signál, a proto je jejich rozlišení závislé na převodníku. [9, 11]

AD převodník transformuje spojitý (analogový) signál na posloupnost čísel v čase (digitální signál). Tato transformace probíhá ve dvou krocích, nejprve se provede vzorkování a pak následuje kvantování. Vzorkování znamená rozdělit spojitý signál na úseky

a z každého úseku vybrat bod, čímž získáme množinu bodů odpovídající vzorkovací frekvenci. Následně se naměřené souřadnice bodů vyjádří číslem, které se zaokrouhlí, protože je k dispozici jen omezený počet desetinných míst. Kvantování tedy znamená nalezení nejbližšího číselného ekvivalentu v dostupné přesnosti. [9, 11]

2.2.3.5. Číslicová regulace

V jednotlivých případech se jedná o snímání hodnot veličin, informujících o stavu procesu, např. teploty, dráhy, tlaku nebo otáček, jejich porovnání s mezními hodnotami a vypočtenými z hodnot řídících veličin a výpočty hodnot nastavovacích signálů. [11]

Regulační smyčka je tvořena regulovanou soustavou a regulátorem. Při realizaci regulátoru počítačovým programem jsou v něm nastavené hodnoty, které ovlivní proces a tím také znovu snímané stavové veličiny. Počítač v tomto systému má za úkol vypočítávat regulační odchylku, tak aby vyhovovala naprogramovaným parametrům regulátoru, vypočítávat nastavovací veličiny a odpovídající signál předávat akčnímu členu v řízeném či regulovaném procesu.

Samooptimalizující (adaptivní) regulační systém může být programovou změnou parametrů upravován a přizpůsobován regulovanému procesu tak, aby byla regulace optimální podle zvolených parametrů. [11]

Schopnost regulátorů se automaticky přizpůsobovat změnám v procesu se nazývá autotuning. Nejprve musí regulační systém získat základní znalost procesu například z přechodové charakteristiky systému, potřebuje získat parametry přenosu nebo hodnoty, ze kterých lze určit parametry regulátoru. Poté je vybrána metoda, která nastaví parametry regulátoru vhodným způsobem a nastartuje regulační pochod. Tento systém je dále optimalizován jednou z následujících metod:

Referenčním modelem - relační smyčka má navíc referenční model, podle kterého se nastavují parametry regulátoru. Tento model ukazuje, jak by měl výstup z regulované soustavy ideálně reagovat na řídící signál. [22]

Duální řízení - jejím cílem je nalezení kompromisu mezi co nejlepším poznáním parametrů regulované soustavy a co nejoptimálnějším řízením. [22]

Samočinně se nastavující regulátor - podle parametrů regulované soustavy se vypočítá regulátor tak, aby byly zajištěny specifikované vlastnosti regulačního obvodu. [22]

Metoda řízeného zesílení - do regulačního obvodu zavedeme pomocnou veličinu, podle které se pak nastavují parametry regulátoru. [22]

2.2.4. Aproximace

2.2.4.1. Avramiho rovnice

Známá jako Johnson-Mehl-Avrami-Kolmogorov nebo JMAK rovnice.

Jedno z praktických využití Avramiho rovnice je při aproximaci přechodové charakteristiky ohřevu pece bez překmitu.
Grafické znázornění Avramiho rovnice se nazývá S křivka. Transformací do Avramiho souřadnic se křivky mění na přímky. [13] Obecný tvar Avramiho rovnice dle [12]:

$$y(t) = 1 - exp(-Kt^n) \tag{2.4}$$

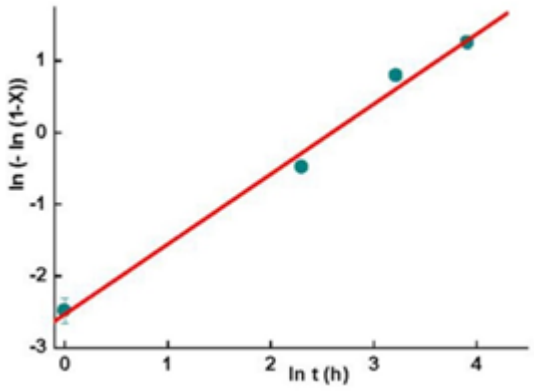
- y(t) regulovaná veličina v čase t
- K (-) konstanta, která udává rychlost ohřevu
- n (-) strmost
- t (s) je čas

Pro výpočet je nutné znát konstanty n a K. Konstanta n se získá po transformaci souřadnic na logaritmické souřadnice. Příklad zjištění konstanty n z rovnice přímky: y = 2,2348x - 15,228, konstanta n se rovná prvnímu členu z rovnice n = 2,2348. Konstanta K je odhad měřeného procesu, získaná z výpočtu K = 1 / celkový čas měření.

Pro transformování exponenciály na přímku je potřeba přepočítat souřadnice na osách a z nich vytvořit graf, který bude vypadat podobně jako graf na na obr.2.7, a z něj zjistit rovnici přímky. Pro přepočet os byl použit vzorec x = ln t pro osu x a pro osu y vzorec y = ln(-ln(1-x).

Výpočet S křivky proběhl dosazením do obecného tvaru Avramiho rovnice. Vypočítaný výsledek byl pouze přibližný, následně se musely doladit velikosti konstant K a n postupným zvyšováním a snižováním jejich hodnot. Změna konstanty K se projevila na délce procesu a při změně konstanty n se měnilo umístění bodů (body se pokládaly na naměřenou křivku).

Při vytváření grafu S křivek se vynesou dosazené hodnoty do obecného tvaru Avramiho rovnice na osu y a na druhou osu se vynese čas.

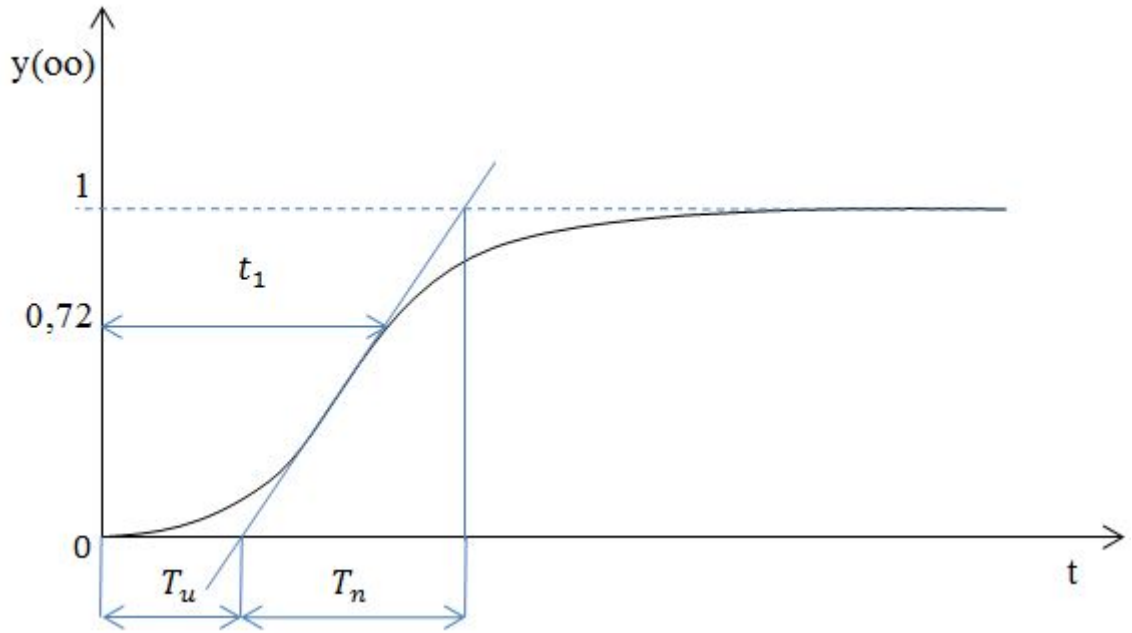


Obr. 2.7.: Avramiho rovnice - transformace exponenciály na přímku.
[14]

Přenosem rozumíme matematický popis chování systému.

2.2.4.2. Strejcova metoda

Slouží ke zjištění přenosu z naměřené přechodové charakteristiky.



Obr. 2.8.: Návod na Strejcovu metodu.

Ideální pro tuto metodu je přepočet naměřených údajů do jednotkového měřítka.

Nejprve odečteme z grafu naměřených údajů doby průtahu a náběhu a z těchto hodnot se vypočte jejich poměr (viz. obr.2.6).

$$\tau = \frac{T_u}{T_n} \tag{2.5}$$

T_u (s) - doba průtahu
T_n (s) - doba náběhu

Určíme případné dopravní zpoždění a pak určíme přenos podle vypočteného τ, protože tato metoda je určena pro dva typy přenosů.

a)$\tau \geq 0{,}104$

$$G(s) = \frac{K}{1 + Ts^n} \tag{2.6}$$

b)$\tau < 0{,}104$

$$G(s) = \frac{K}{(1 + T_1 s)(1 + T_2 s)} \tag{2.7}$$

T_i (s) - čas
V případě za a) se určují konstanty z tabulky:

n	1	2	3	4	5	6	7	8	9	10
τ (s)	0	0,104	0,218	0,319	0,410	0,493	0,570	0,642	0,709	0,771

Tabulka 2.2.: Tabulka konstant pro Strejcovu metodu.

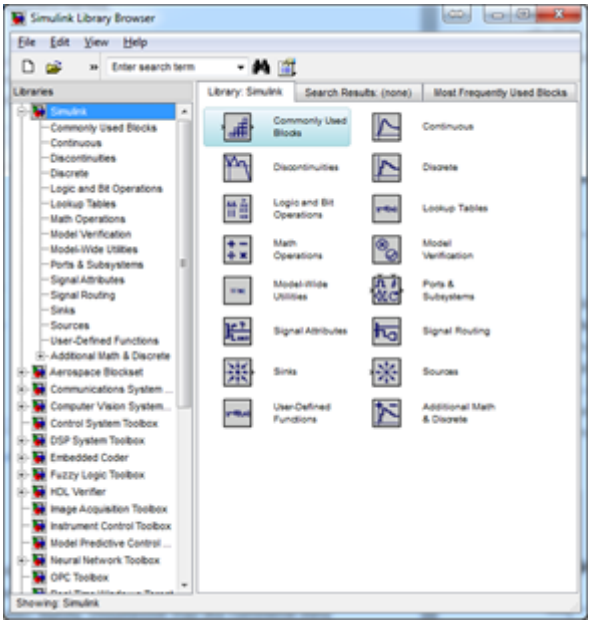
τ_2	0,1	0,2	0,3	0,4	0,5	0,6
τ (s)	0,050	0,072	0,084	0,092	0,097	0,100

Tabulka 2.3.: Tabulka pro určení konstanty τ_2.

Po zjištění konstant vypočteme velikosti T_1 a T_2 po dosazení vypočítaných hodnot do vzorců výše. A konstanta K se rovná 1 po přepočtu do jednotkového měřítka.

2.2.5. Matlab - Simulink

Prostředí Simulink je nadstavbou Matlabu a slouží k simulaci a modelování dynamických systémů. Vyžívá se algoritmů z Matlabu pro numerické řešení nelineárních diferenciálních rovnic. V tomto prostředí se modely vytváří pomocí blokových schémat. [20]

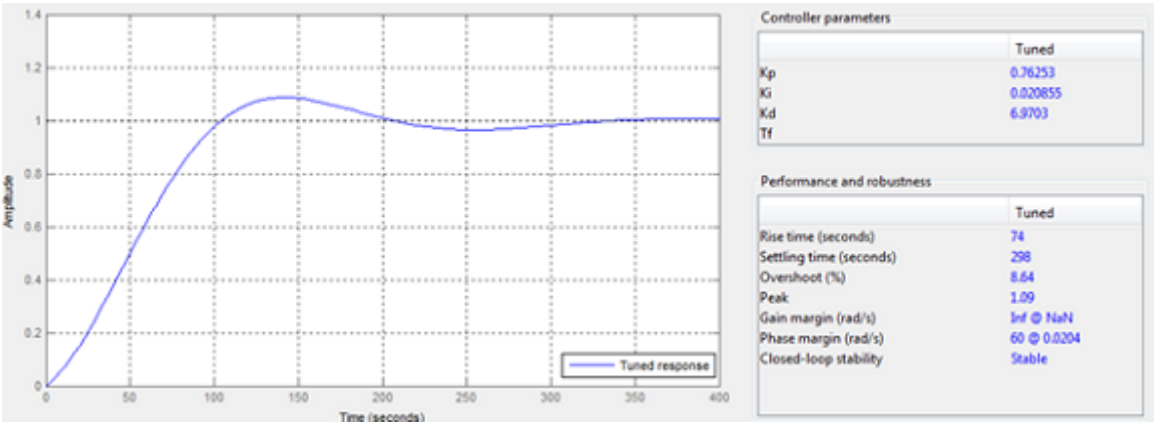


Obr. 2.9.: Úvodní obrazovka při spuštění nadstavby Matlab - Simulink.

Na úvodní obrazovce Simulinku (obr.2.9) je vidět vlevo panel knihoven, vpravo je panel nástrojů dostupných ve vybrané knihovně. Použití jednotlivých bloků se provede jen přetažením do pracovního prostoru a následným pospojováním.

2.2.6. Matlab - Pidtool

Tento nástroj slouží na zpracování zjištěného přenosu. Po zadání přenosu a typu regulátoru ihned vykreslí křivku, která se dá dále upravit. Vypočtené parametry regulátoru jsou k dispozici po použití tlačítka show parameters (zobraz parametry). Pro správné použití tohoto je potřebné zadat přenos, ze kterého se vypočítají parametry regulátoru. Jak je vidět na obr.2.10, že tento nástroj pracuje s křivkou přepočítanou do jednotkového měřítka.



Obr. 2.10.: Návrh regulátoru v pidtool.

3. Použité přístroje

V této kapitole jsou uvedeny přístroje použité při měření na pecích a při realizaci regulátoru na Netduinu Plus 2.

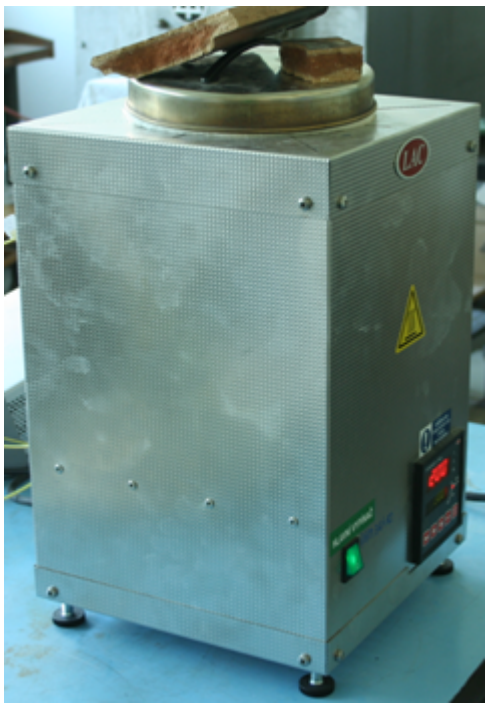
3.1. Použité přístroje pro měření

Pec LAC:
V tabulce tab.3.1 jsou uvedeny některé údaje opsané ze štítku pece.

Tabulka 3.1.: Štítkové údaje pece LAC.

LMV 2112	v.č. 1210692
Jmenovité napětí 1/N/PE/ 230 V AC 50	Hz Přípojný příkon 3,7 kVA
Jmenovitý topný příkon 1,8 kW	Jmenovitá teplota 1200°C
Jmenovitý proud 7,8 A	Hmotnost 15 kg
Stupeň krytí IP40	Rok výroby 2012

Na obr.3.1 je vidět pec LAC s regulátorem Ht40 AL při měření.



Obr. 3.1.: Pec LAC s Ht40 AL při měření. Položené desky na víku slouží jako závaží proti odklopení víka.

Pec ESA:
Údaje pece jsou opsány ze štítku pece v tabulce tab.3.2.

Tabulka 3.2.: Štítkové údaje pece ESA.

Typ K59	v.č. 372/1988
Topný příkon 750 W	Jmenovitá teplota 1000°C
Stupeň krytí IP20	Jmenovité napětí 220 V AC 40 - 60 Hz

Měřící ústředna Agilent 34972A:

- Lan, USB
- 3 sloty na karty
- Paměť 50k na naměřené hodnoty
- Rozhranní RS 232, GPIB

Ústředna je vhodná pro záznam dat, přepínání a použití v automatizovaných testovacích systémech. Tato ústředna je vybavena softwarem Agilent BenchLink Data Logger. Při měření na pecích byla data z měření ukládána na flash disk.

Modul 34902A k měřící ústředně:

- 16 kanálový multiplexer
- rychlost přepínání 250 kanálů/s
- šířka pásma 10 MHz
- vestavěná teplotní reference termočlánku
- Přesnost při měření termočlánky:
- J: -150 – 1200°C je 1°C
- K: -100 – 1200°C je 1°C

Regulátor pece LAC Ht40 AL:

- Regulátor umožnuje jednoduchou programovou regulaci
- Umožnuje náběh a výdrž
- Regulaci na konstantní hodnotu
- V regulátoru lze nastavit dvoupolohovou regulaci nebo PID regulaci pro topení
- Přesnost ± 0,1 % z rozsahu
- Teplotní stabilita $\pm\ 0,1$°C teploty okolí
- Napěťová stabilita ± 0,1 % změny napájecího napětí
- Hranice parametrů od -499°C až 2499°C

Na obr.3.1 je zobrazen regulátor Ht40 AL.

Obr. 3.2.: Regulátor Ht40 AL. [24]

Přiřazení termočlánků k jednotlivým kanálům při měření na peci LAC je v tab.3.1.

Tabulka 3.3.: Přiřazení termočlánků na peci LAC.

Typ K59	v.č. 372/1988
Kanál	Označení termočlánku
01	IRCO-010
02	CHAL-010

Přiřazení termočlánků k jednotlivým kanálům při měření na peci ESA.

Tabulka 3.4.: Přiřazení termočlánků na peci ESA.

Typ K59	v.č. 372/1988
Kanál	Označení termočlánku
01	CHAL-032
02	IRCO-010

Tabulka 3.5.: Použité termočlánky.

Typ	Označení	Průměr	Délka	Rozsah (°C)	výrobce
J	IRCO-010	0,1"	12"	-210 až 1200	OMEGA Engineering
K	CHAL-010	0,1"	12"	-270 až 1372	OMEGA Engineering
K	CHAL-032	0,32"	12"	-270 až 1372	OMEGA Engineering

3.2. Použité prvky při realizaci regulátoru

Relé KSD240AC8 dle [15]:

- SSR relé pro montáž na chladič 40A/600V, izolační pevnost 4000 V.
- Spínání v nule: ANO
- Spínání (Ovládání/Zátěž): DC/AC
- Ovládací napětí: 4..32 V
- Izolační pevnost: 4000 V
- Trvalý spínaný proud: 40 A
- Špičkový spínaný proud: 400 A
- Jmenovité napětí zátěže: 250 V AC
- Špičkové spínané napětí: 600 V
- Pracovní frekvence zátěže: 47 - 70 Hz
- Ton/off: 8,3/8,3 ms
- Provozní teplota: -30..+100°C
- Výrobce: COSMO ELECTRONIC CORPORATION

Obr. 3.3.: SSR relé. [15]

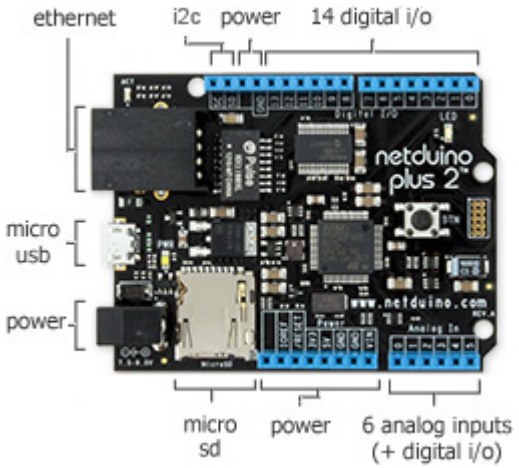
Modul na měření teploty MOD-TC dle [16]:

- Měření teploty
- Používá termočlánek typ K
- Komunikuje přes SPI
- Rozmezí teplot: 0 – 1023.75°C
- Přesnost: $0,25$°C
- Součásti: Konektor připojení termočlánku, UEXT konektor
- Rozměry: 22 x 18 mm

Obr. 3.4.: MOD - TC. [16]

Netduino Plus 2 dle [17]:

- STMicro 32 -bitový mikrokontrolér
- Rychlost: 168MHz , Cortex - M4
- Interní paměť: 384 KB
- RAM: 100+ KB
- 10 mbps Ethernet
- Podpora Micro SD karty až 2 GB
- Vstupní napětí: 7,5-9,0 V DC nebo USB napájením
- 25 mA na Pin max. proud
- Digitální I/O je 3,3 V a je možno až 5 V



Obr. 3.5.: Netduino Plus 2. [17]

FTDI Basic Breakout dle [19]:

- Modul na posílání dat do PC
- Vytváří virtuální sériový port v PC
- 6 pinový konektor (ze spodu)
- 3,3V nebo 5V
- Indikace komunikace pomocí TX a RX diod

Obr. 3.6.: FTDI Basic Breakout. [19]

SPI (Serial Peripheral Interface)
Je sériové periferní rozhraní. Používá se pro komunikaci mezi řídícími mikroprocesory a ostatními integrovanými obvody (EEPROM, A/D převodníky, displeje...). Komunikace je realizována pomocí společné sběrnice, full duplex a probíhá v Master / Slave módu. [18]

Adresace se provádí pomocí zvláštních vodičů, které při logické nule aktivují příjem a vysílání zvoleného zařízení (piny $\overline{SS}$ nebo $\overline{CS}$). [18]

Průběh komunikace:

Pro komunikaci Master nastaví log. 0 na $\overline{SS}$ zařízení, se kterým chce komunikovat. [18]

Pak začne generovat hodinový signál na SCLK a v té chvíli vyšlou obě zařízení svoje data, přičemž MOSI (Master Out, Slave In) je vždy Master výstup, Slave vstup a MISO (Master In, Slave Out) je Master Vstup, Slave výstup. [18]

Jakmile jsou data vyslána, může komunikace dále pokračovat:

Master dále dodává hodinový signál, hodnota $\overline{SS}$ se nemění nebo může být ukončena: Master přestane vysílat hodinový signál a nastaví $\overline{SS}$ do log. 1. [18]

Délka vyslaných dat je 8bit (Byte) a nebo 16bit (Word).

3.2.1. Průběh měření

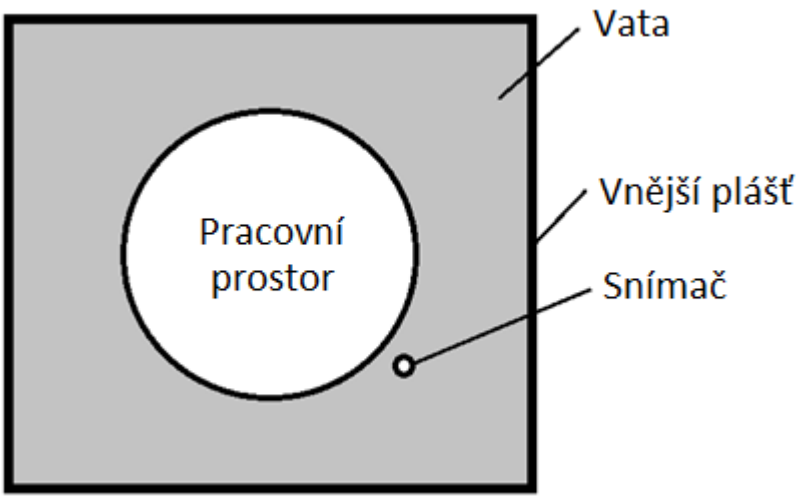
Termočlánky J a K byli vloženy do pece, jak je zobrazeno na obr.3.11. K článek byl volně zavěšen v prostoru a J článek byl zavěšen u stěny vnitřního válcového prostoru pece. Oba termočlánky byly umístěny v úrovni topné spirály. V případě umístění pod úrovní topné spirály byly výsledky měření zkresleny jejich umístěním. Toto bylo vidět i při prvním testovacím (přípravném) měření do 200°C.

Obr. 3.7.: Fotka zapojení z měření na peci LAC.

Na obr.3.7 je vidět průběh měření na peci LAC. V popředí je měřící ústředna, do které je zapojeno prodlužovací kompenzační vedení ze stejného materiálu, na kterém jsou připojeny termočlánky J a K umístěné v pracovním prostoru pece. Na víko pece musela být umístěna izolace chránící před vlivem okolního prostředí. Termočlánky jsou navlečené v ochranných keramických vložkách proti dotyku, které byly dále odizolovány izolační vatou od konstrukce pece.

Během všech měření byla zaznamenávána teplota na termočlánku J a teplota zobrazená na peci LAC, která je vybavena zobrazovacím displejem regulátoru na rozdíl od pece ESA.

Problém při měření teploty pece představuje umístění jejího snímače teploty (obr.3.8). Může být umístěn způsobem takovým, že se buď přímo dotýká vnějšího pláště pracovního prostoru, nebo je mezi snímačem a pracovním prostorem vzduchová mezera. Vzduchová mezera při takovém umístění působí jako izolant, tudíž pak je naměřená teplota nižší než ve skutečnosti.

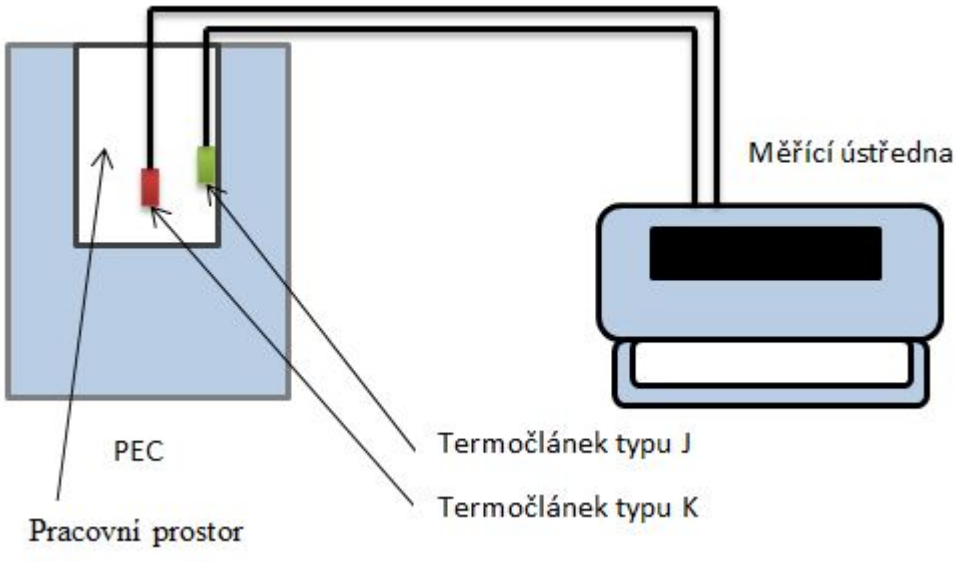


Obr. 3.8.: Schéma pracovního prostoru pece.

Průběh měření na peci ESA byl podobný měření na předchozí peci jak je vidět na obr.3.9. Jediný rozdíl spočíval v tom, že byla měřena charakteristika ohřevu pece bez regulátoru. Tato pec není vybavena vlastním měřícím zařízením, proto nebylo možné porovnávat teplotu naměřenou pecí samotnou a měřící ústřednou. Vzhledem ke stáří pece a obavám před jejím možným poškozením bylo měření ukončeno ještě před dosažením ustáleného stavu. Z tohoto důvodu není naměřena celá charakteristika.

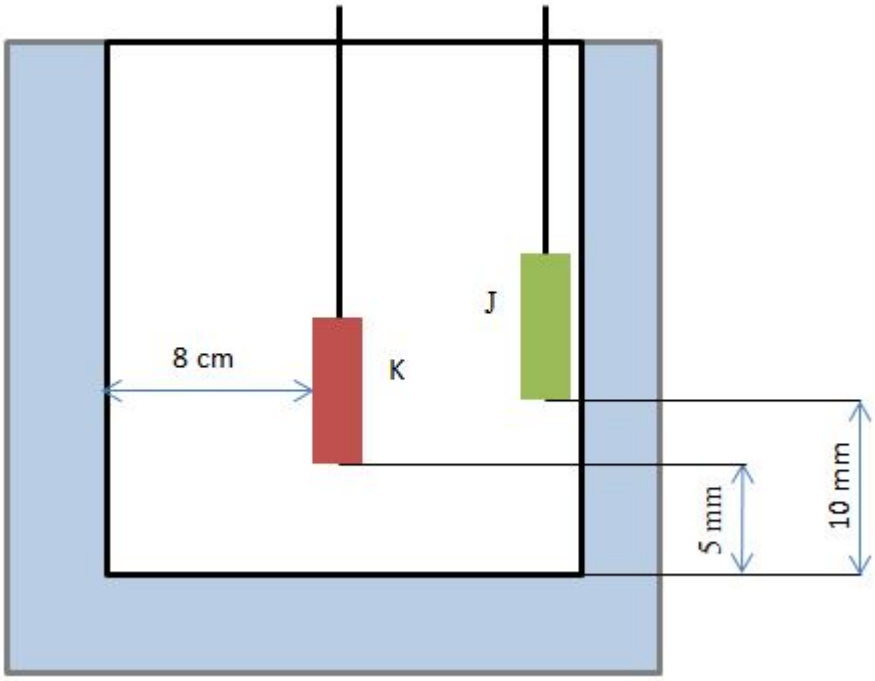
Obr. 3.9.: Měření na peci ESA.

Pec ESA má při měření tu výhodu, že se ve víku, které kryje pracovní prostor, nachází dva otvory vhodné pro zavedení termočlánků. Pro měření byl použit termočlánek typu K, který je uveden v tabulce termočlánků tab.3.5, ve variantě s větším průměrem a nastaveným měřením po 5 sekundách. S ohledem na rychlost ohřevu pece a celkovou dobu tohoto ohřevu je doba 5 sekund mezi měřenými hodnotami teplot dostačující.



Obr. 3.10.: Schéma zapojení měření.

Schéma zapojení měření (obr.3.10) je pro měření na obou pecích totožné. Byla použita stejná měřící ústředna typu 34972A a stejné umístění termočlánků v pracovním prostoru pece. Ze schématu je vidět, že termočlánek typu K je umístěn volně v prostoru a termočlánek typu J se dotýká stěny pracovního prostoru.



Obr. 3.11.: Umístění termočlánků v pracovním prostoru pece.

Na obrázku (obr.3.11) je vidět umístění termočlánků uvnitř pracovního prostoru pece. Termočlánek typu K byl umístěn uprostřed prostoru pece. Druhý termočlánek typu J se dotýkal stěny v polovině její výšky.

4. Aproximace a návrh regulátoru

V této kapitole se nachází prezentace výsledků aproximace naměřených křivek na peci LAC. Dále jsou zde naměřený graf hodnot z pece ESA a návrh regulátoru v programu Matlab.

4.1. Aproximace naměřených křivek a výpočet S křivek

V tabulce (tab.4.1) jsou uvedeny vypočítané konstanty a následně dosazené do Avramiho rovnice pro každé měření a použitý termočlánek.

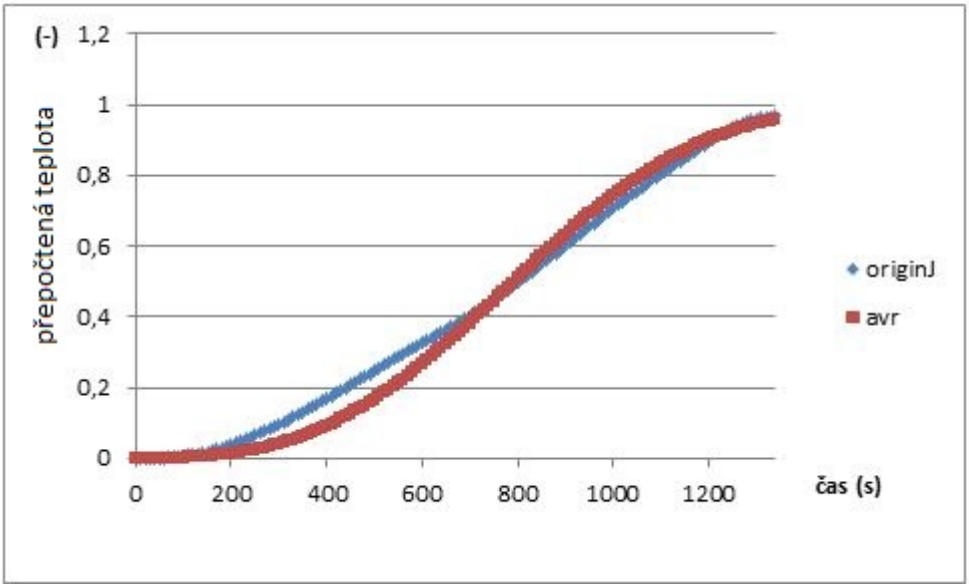
Tabulka 4.1.: Údaje k naměřeným a aproximovaným křivkám.

Termo-článek	**Avramiho rovnice**	**Platnost rovnice (s)**	**Čas ustálení (s) ± 5%**	**Hodnota maximálního překmitu** (°C)
J	$y_{200}(t) = 1 - e^{-1,3x10^{-9}t^{3,42}}$	520	1700	61,013
K	$y_{200}(t) = 1 - e^{-5,9x10^{-9}t^{3}}$	810	2000	50,029
J	$y_{300}(t) = 1 - e^{-5,9x10^{-10}t^{3,382}}$	740	1550	58,947
K	$y_{300}(t) = 1 - e^{-5,5x10^{-9}t^{2,92}}$	950	1790	49,19
J	$y_{400}(t) = 1 - e^{-5,5x10^{-9}t^{2,95}}$	880	1850	67,945
K	$y_{400}(t) = 1 - e^{-5,5x10^{-9}t^{2,85}}$	1010	1840	58,46
J	$y_{500}(t) = 1 - e^{-1,1x10^{-9}t^{3,1}}$	1040	1960	58,013
K	$y_{500}(t) = 1 - e^{-2,5x10^{-9}t^{2,92}}$	1130	1830	50,863
J	$y_{600}(t) = 1 - e^{-3,2x10^{-9}t^{2,9}}$	1250	1220	40,796
J	$y_{700}(t) = 1 - e^{-2,7x10^{-9}t^{2,9}}$	1340	1300	31,19
K	$y_{700}(t) = 1 - e^{-7,2x10^{-11}t^{3,4}}$	1400	1430	30,39

V následující tabulce (tab.4.2) je uveden čas potřebný k dosažení určité procentní úrovně z cílové teploty, na kterou se měla pec zahřát. Údaje jsou z naměřených hodnot termočlánkem typu J a aproximované hodnoty Avramiho rovnicí (v grafech a tabulce značeno avr).

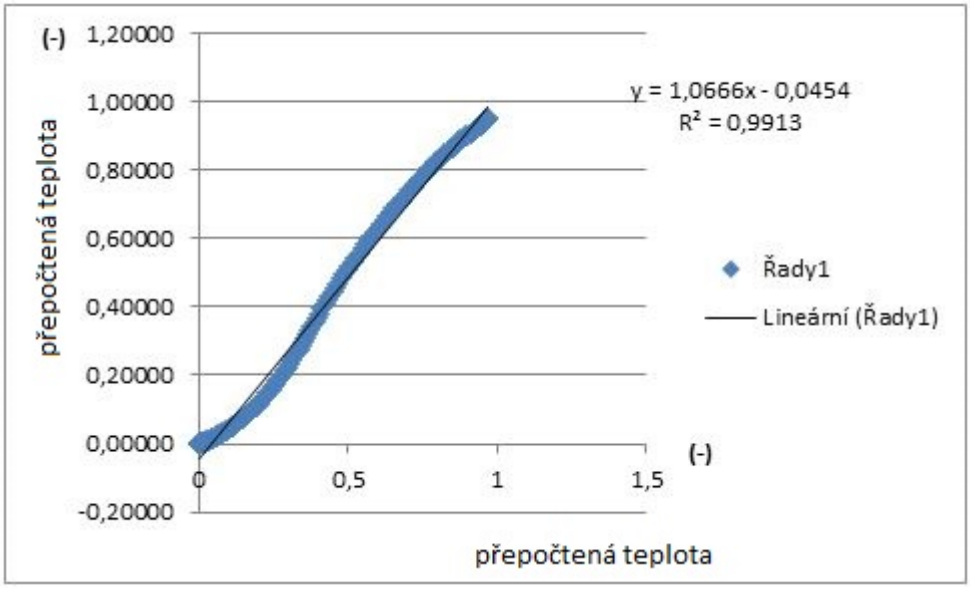
Tabulka 4.2.: Hodnoty u termočlánků typu J.

	63,2%		**90%**		**90%**	
(°C)	**J**	**avr**	**J**	**avr**	**J**	**avr**
200	380	390	490	510	510	540
300	520	530	680	690	710	740
400	610	620	830	840	860	910
500	760	770	970	1000	1000	1070
600	840	840	1080	1140	1130	1250
700	910	880	1210	1190	1290	1340



Obr. 4.1.: Příklad výsledku aproximace s použitím Avramiho rovnice pro termočlánek typu J s měřenou teplotou na 700°C. (originJ = naměřená data termočlánkem typu J, avr = aproximovaná data Avramiho rovnicí)

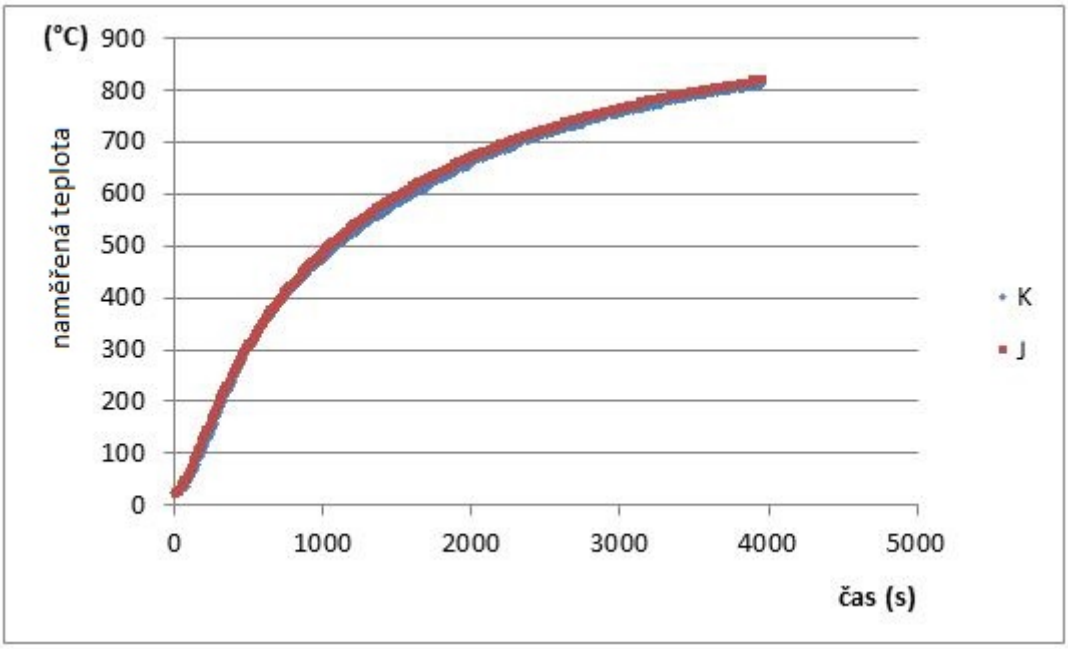
Z grafu na obr.4.1 je vidět jak křivka aproximovaná Avramiho rovnicí kopíruje tvar křivky naměřené. Z počátku se obě křivky shodují, ale v čase v 200 s se křivky rozchází a přiblíží se na skoro shodný tvar opět se spojí až v čase 700 s.



Obr. 4.2.: Porovnání naměřených a aproximovaných hodnot Avramiho rovnicí pro termočlánek typu J s měřenou teplotou na 700°C.

Na obr.4.2 je na grafu znázorněno, jak je křivka naměřená podobná aproximované křivce. Na tomto grafu je vidět odlišnost obou křivek, podle porovnání shody bodů u obou křivek. Kdy se porovnávají jen souřadnice bodů osy y, na které jsou přepočítány hodnoty naměřených stupňů Celsia do jednotkového měřítka.

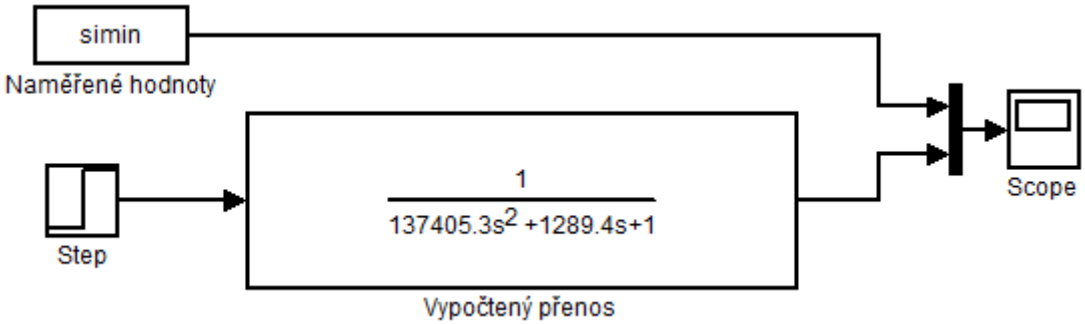
4.2. Naměřené hodnoty



Obr. 4.3.: Naměřené hodnoty na peci ESA termočlánky J a K.

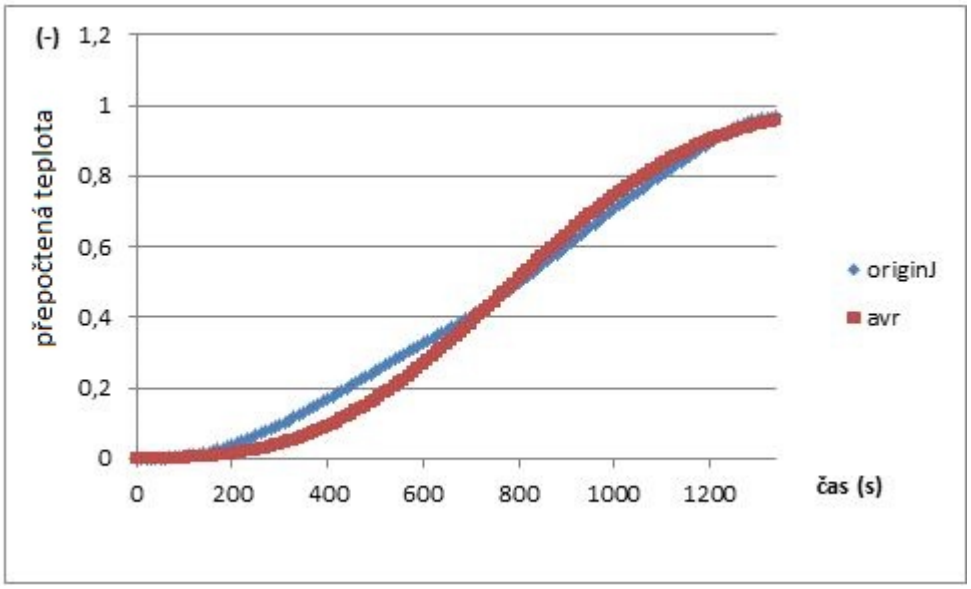
Na grafu (obr.4.3) je vidět, že měření přechodové charakteristiky pece ESA nebylo ukončeno v ustáleném stavu. Při ukončení měření bylo dosažení teploty 820°C na termočlánku typu K a 810°C typu J. Podle naměřených křivek se dá říci, že oba termočlánky naměřily téměř shodnou křivku.

4.3. Matlab - Simulink



Obr. 4.4.: Ukázka simulace odezvy na jednotkový skok v Matlabu.

V schématu odezvy na jednotkový skok představuje blok simin naměřená data přepočítána do jednotkového měřítka vložená do Matlabu. Blok Scope představuje výstup do grafu a Step realizuje skok mezi dvěma nadefinovanými hladinami v určitém čase.



Obr. 4.5.: Výsledek odezvy na jednotkový skok pro 800°C. v Matlabu (červená čára představuje naměřené hodnoty a modrá odezvu na jednotkový skok podle vypočítaného přenosu).

Na grafu v obr.4.5 je vidět odezvu na vypočítaný přenos pro teplotu 800°C. Vypočtený přenos byl vypočítán pro 1000°C, ale výsledek neseděl na naměřené křivce. Následně byl přenos vypočítán pro další teploty a opět vyzkoušen jak moc sedí na naměřenou křivku v Matlabu. Graf z Matlabu byl vyexportován do Excelu.

Vybrané hodnoty: 950°C, 900°C, 850°C, 840°C, 800°C. Pro každou z těchto hodnot byl spočítán přenos, poté byl vybrán přenos pro hodnotu 800°C a jeho výsledek je vidět na obr.4.5. V tomto případě je dopravní zpoždění velmi malé, proto ho neuvažujeme. Vypočtený přenos byl dosazen do rovnice 2.7.

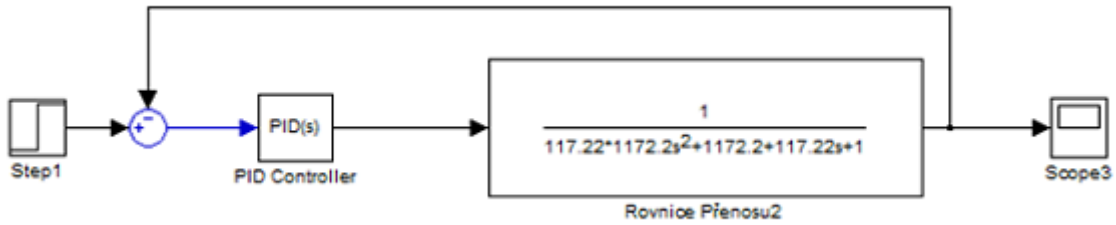
V následující tabulce jsou uvedeny vypočítané konstanty přenosu po převedení původních hodnot do jednotkového měřítka.

Tabulka 4.3.: Tabulka vypočítaných konstant pro přenos.

Teplota (°C)	Zesílení	0,72 y	Časová konstanta T_1 (s)	Časová konstanta T_2 (s)
1000	1	2885	2087,5	208,75
950	1	2410	1743,8	174,38
900	1	210	1772,75	177,27
850	1	1985	1526,7	152,67
840	1	1780	1287,9	128,8
800	1	1620	1172,2	117,22

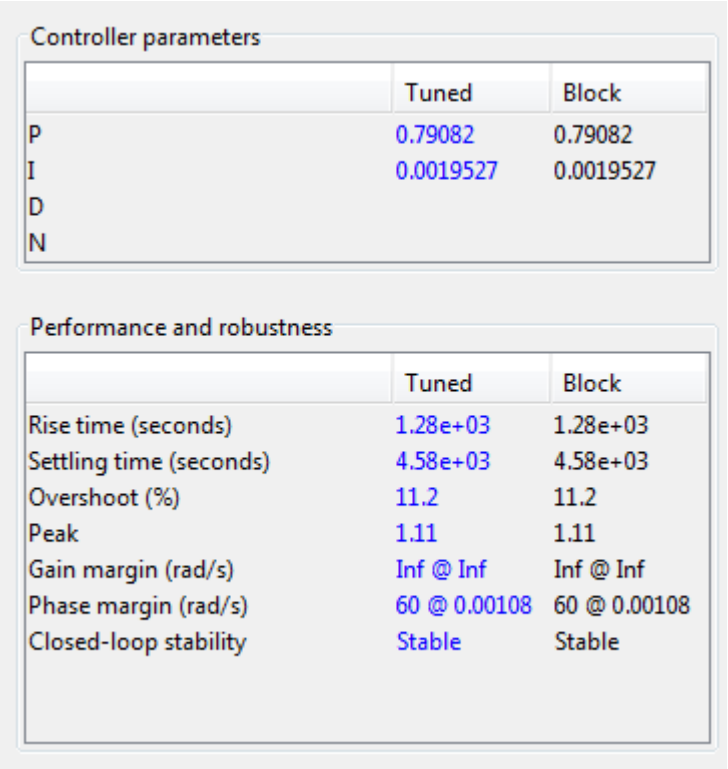
4.3.1. Naměřené hodnoty

Po použití výpočetního nástroje byly dosazeny vypočtené hodnoty do simulace v simulinku (PIDtool je to samé jako PIDtuner).



Obr. 4.6.: PI regulátor namodelovaný v Matlab - Simulink.

Na obr.4.7 jsou uvedeny vypočítané parametry nástrojem PIDtool, v horní části jsou parametry regulátoru, hodnoty pro jednotlivé složky. V další části obrázku jsou vypsány dodatečné parametry (například překmit, stabilita, atd.).



Controller parameters

	Tuned	Block
P	0.79082	0.79082
I	0.0019527	0.0019527
D		
N		

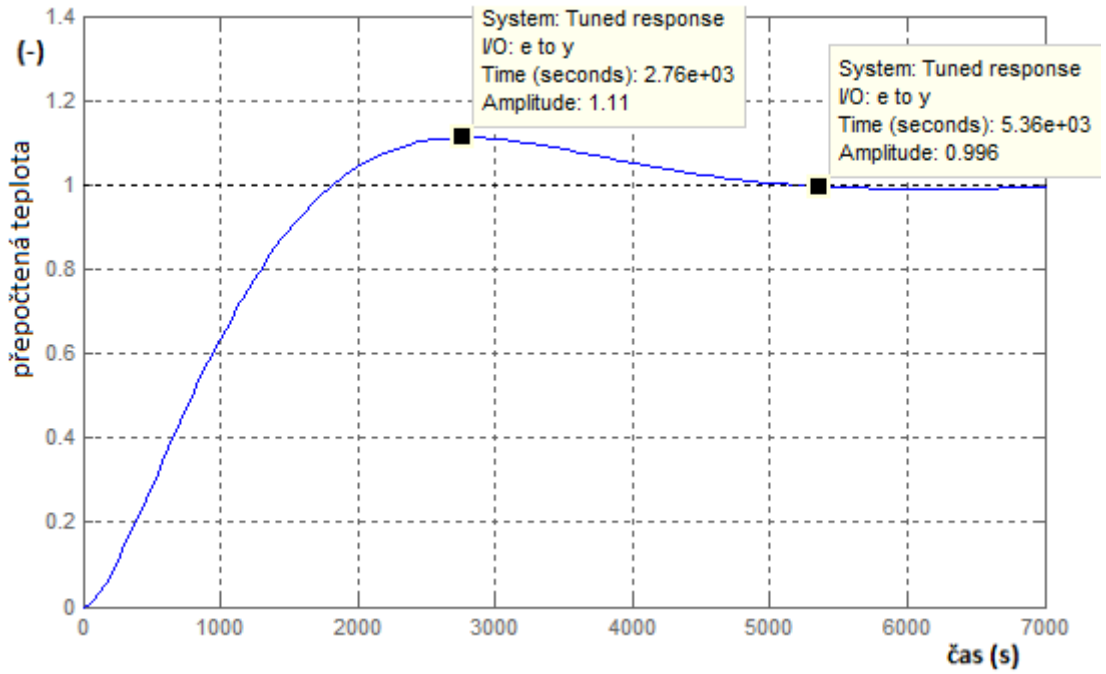
Performance and robustness

	Tuned	Block
Rise time (seconds)	1.28e+03	1.28e+03
Settling time (seconds)	4.58e+03	4.58e+03
Overshoot (%)	11.2	11.2
Peak	1.11	1.11
Gain margin (rad/s)	Inf @ Inf	Inf @ Inf
Phase margin (rad/s)	60 @ 0.00108	60 @ 0.00108
Closed-loop stability	Stable	Stable

Obr. 4.7.: Vypočítané parametry od PIDtool.

Na následujícím grafu (obr.4.8) je vidět výsledek regulace při použití vypočteného přenosu s použitím PID Tune. Hodnoty na ose y jsou přepočítány do jednotkového měřítka, hodnota 1 odpovídá teplotě 800°C.

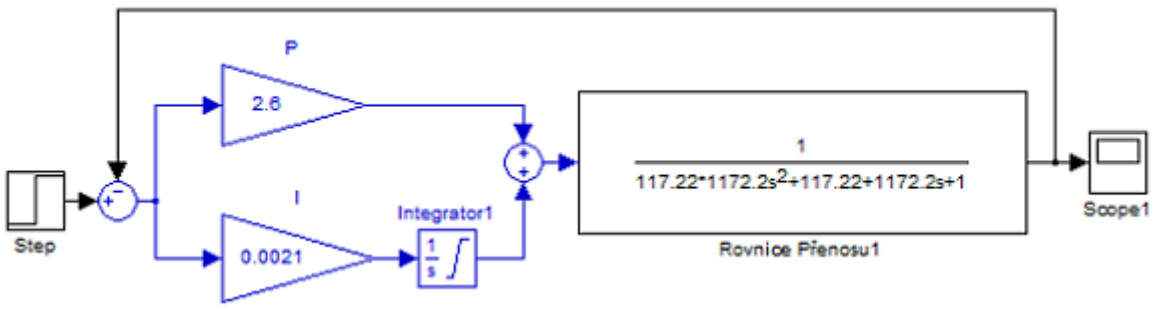
Na obrázku (obr.4.8) je zobrazen výsledek simulace pomocí regulátoru PI přímo v nástroji Matlabu PIDtool.



Obr. 4.8.: Výsledek regulace s dodatečnými parametry.

Na obrázku (obr.4.8) jsou vyznačeny dva body, které zobrazují maximální hodnotu překmitu a druhý kdy se systém ustálí. Ustálení systému znamená dosáhnout ±5 % požadované hodnoty. Regulovaný systém při regulaci PI regulátorem má překmit 11,2 % a k ustálení soustavy dojde po 5360 s.

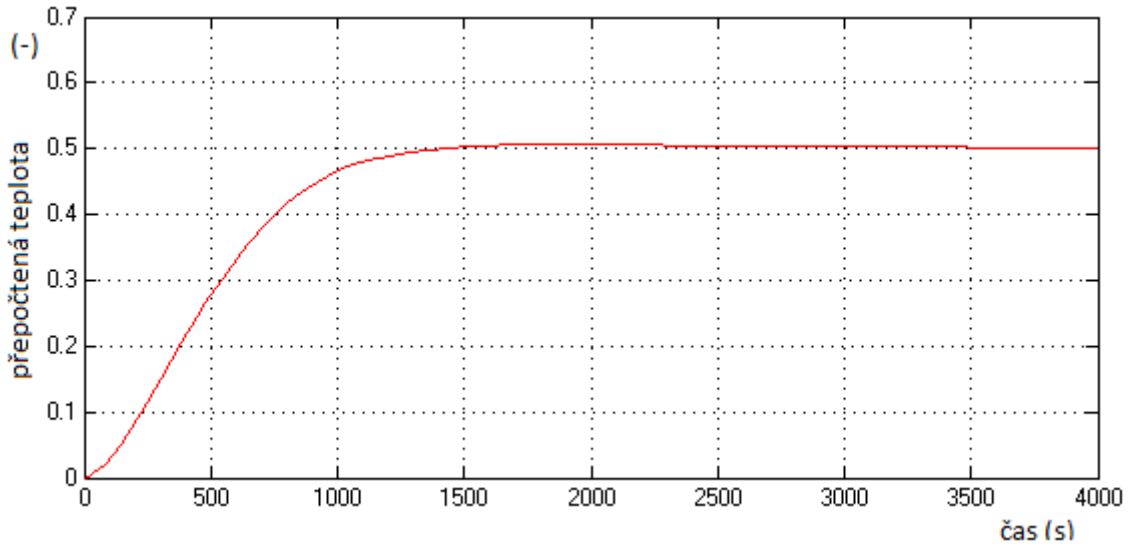
PIDtool má nevýhodu nastavení, špatně se nastavuje hodnota překmitu, proto byla sestavena druhá varianta systému (obr.4.9) pro simulaci bez použití bloku PID.



Obr. 4.9.: PI regulátor bez PID bloku v prostředí Matlab - Simulink.

Blok s názvem P je propociální složka a blok I s integrátorem společně tvoří integrační složku regulátoru PI. V tomto byly použity hodnoty z PIDtool, jako výchozí hodnoty. Pro stanovaný 5 % překmit, ale byli nedostačující. Z tohoto důvodu byli honoty manuálně měněny, dokud nebyl dosažen co nejmenší překmit.

Postup pro manuální nastavení regulátoru spočíval ve vynulování hodnoty I a nastavení složky P dokud nebyl na výstupu v grafu malý překmit. Nalezená hodnota pro složku P byla 3,2, původní hodnota 0,79. Následně byla nastavena hodnota P na 75 % a postupně se měnila složka I, která z původní hodnoty 0,0019 se měnila na hodnotu 0,0021. Po úpravě složky P na hodnotu 2,6 bylo dosaženo výsledku regulace, který je zobrazen na obr.4.10.

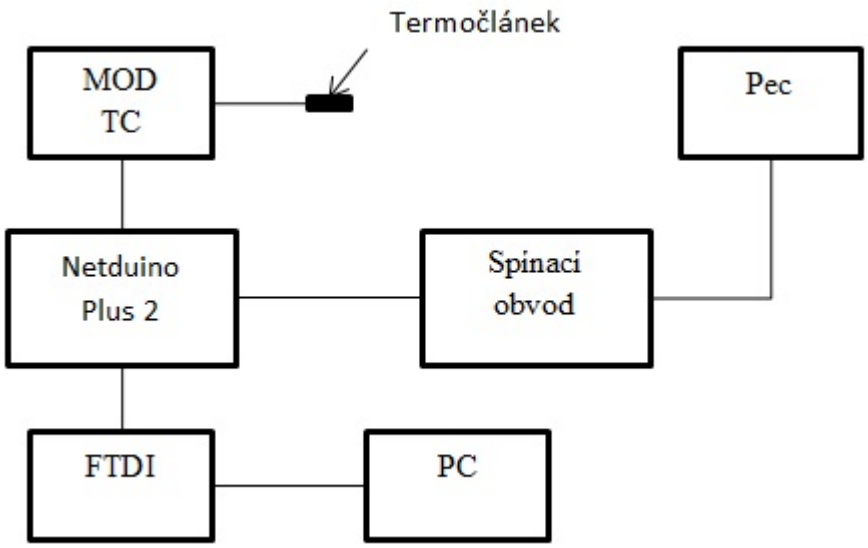


Obr. 4.10.: Výsledek regulace s ručně měněnými parametry P a I.

V grafu na obr.4.10 je stupnice v jednotkovém měřítku, kdy 1 je 800°C.
Výsledné konstanty z této simulace byli dosazeny do programu regulátoru.

5. Realizace regulátoru

V této kapitole je popsána realizace hardwarové části regulačního obvodu a jsou zde popsány vybrané části programu z netduina a aplikace v počítači. Pro psaní kódu bylo použito Microsoft Visual Studio 2010. Na následujícím obr.5.1 je zjednodušené schéma zapojení celého obvodu.



Obr. 5.1.: Zjednodušené schéma zapojení řídícího obvodu.

Ve schématu je vidět, že počítač a netduino jsou propojeny FTDI modulem, který vytváří virtuální sériový port. Na netduino je připojen modul MOD - TC, na který je připojen termočlánek, který měří teplotu v peci. Aby byl nákres přehlednější, je termočlánek nakreslen mimo pec. Spínací obvod je napojen na netduino, které ho řídí, a tento obvod ovládá topení pece.

Celý obvod pracuje tak, že se pošle řídící příkaz z počítače, ve kterém je spuštěna aplikace na řízení nedtuina, pro spuštění nebo vypnutí celého procesu regulace. Když je příkaz na spuštění procesu poslán, tak se začnou vysílat naměřená data z připojeného termočlánku do netduina a z něj po zpracování dat do aplikace v počítači, kde jsou vypisována do tabulky. Řízení pece se uskutečňuje PI regulátorem s PWM cyklem, který je re-

alizován netduinem, na které je připojen spínací obvod, který řídí topení pece. V naprogramovaném PI regulátoru se porovnává aktuální hodnota teploty naměřená v peci a zadaná hodnota, na kterou se reguluje a tím se řídí PWM cyklus.

Kód programu použitého v netduinu i v aplikaci v počítači je vložen jako příloha.

5.0.1. Programování Netduina

V případě použití v realizaci programu řízení pece v jazyce C#, v prostředí Microsoft Visual Studio s použitím na Netduino Plus 2 je použito i šablon (netduino SDK), které mají už vložené potřebné knihovny použité pro netduino. Celý kód je přiložen jako příloha A. V následujícím textu je uvedena zjednodušená struktura kódu.

Struktura programu:

- použité knihovny
- deklarace globálních proměnných a nastavení
- hlavní část programu
- vlákno programu (realizace PWM)
- přerušení programu (čtení sériového portu)

Použité knihovny jsou nezbytnou součástí programu, protože zpřístupňují příkazy na ovládání hardwarových komponent netduina. Především knihovny od SecretLabs se věnují ovládání hardwaru a System.IO obsahuje příkazy k ovládání vstupů a výstupů u netduina.

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using SecretLabs.NETMF.Hardware;
using SecretLabs.NETMF.Hardware.Netduino;
using System.Text;
using System.IO;
using System.IO.Ports;
```

V hlavní části programu se na jeho začátku nachází deklarace globálně proměnných použitých v celém programu. Podstatná je z nich proměnná (zadanaTeplota) pro základní nastavení zadaní teploty regulace, kdy je zadána požadovaná hodnota, na kterou se bude pec ohřívat. Podobnou funkci jako předchozí má i hodnotaKi, nastavení složky I u PI regulátoru a nastavenaPWM sloužící k nastavení délky cyklu u PWM. Další důležitá proměnná (serial) je deklarace sériového portu pro komunikaci po sériové lince, kdy je například potřeba nadefinovat konkrétní port a rychlost.

```
//nastaveni PWM, strida
        //0.0 = 0%, 1.0 = 100%
        static float nastavenaPWM = 0.0f;
        //pozadovana teplota ve stupnich celsia, 0 znamena vypnuto
        static int zadanaTeplota=100;
        static bool mamVysilat = true;
```

Inicializace nové instance třídy SerialPort pomocí zadaného názvu portu, přenosové rychlosti, paritního bitu, počtu datových bitů a stop bitu.

```
   //deklarace serioveho portu
        static SerialPort serial = new SerialPort(SerialPorts.COM1, 9600, Parity.No
        //promenna pro cteni prichoziho retezce
        static string lineFin = "";
        //promenna pro PI regulator
        static double hodnotaKi = 0;
```

Nastavení SPI portu realizuje komunikaci mezi netduinem a modulem MOD - TC, který posílá data netduinu a v něm se pak zpracovávají. V následujícím kódu jsou uvedeny nastavovací parametry.

```
//nastaveni portu SPI pro pripojeni termoclanku
        SPI.Configuration Device1 = new SPI.Configuration(
        Pins.GPIO_PIN_D10, // vybrany pin
        false,             // aktivni stav v nule
        0,                 // nastaveni casu portu
        0,                 // doba setrvani na portu
        true,              // klidovy stav hodin
        true,              // vzorkovaci cas okraje
        250,               // taktovací frekvence v KHz
        SPI_Devices.SPI1   // pouzita sbernice
        );
```

Nastavené parametry SPI komunikace se používají nadefinováním proměnné, která tyto parametry používá. V tomto případě je to proměnná SPIBus.

```
//deklarace promenne pouzivajici SPI
        SPI SPIBus = new SPI(Device1);
        SPIBus.Config = Device1;
```

V programu je přerušení, které se spouští při zavolání funkce serial_DataReceived. V tomto přerušení se nachází funkce, která kontroluje příchozí řetězec znaků (string) po sériové lince do netduina. Příchozí řetězec se čte po jednom znaku, dokud nepřijde středník, podle toho funkce pozná, že se jedná o řídící příkaz.

```
//cteni prichozich znaku ze serioveho portu
        cteciByte = (char) serial.ReadByte();

        //cte se seriovy port dokud neprijde znak nesplnujici podminku
        if (cteciByte != ';')
        {
           lineFin += cteciByte;
        }
```

Když takový příkaz se středníkem přijde, tak se poté kontroluje co obsahuje, respektive co má program udělat. Nadefinované příkazy jsou pouze dva, start, zapnutí vysílání dat a nebo stop, kdy se ukončí vysílání dat po sériové lince.

```
    else
              {
                  if (lineFin.CompareTo("start") == 0)
                  {
                      mamVysilat = true;
                      Debug.Print("start programu");
                  }
                  else if (lineFin.CompareTo("stop") == 0)
                  {
                      mamVysilat = false;
                      zadanaTeplota = 0;
                      Debug.Print("konec programu");
                  }
                  //cisteni promenne po kazdem pruchodu cyklu
                  lineFin = "";
              }
```

Na konci cyklu se proměnná na čtení řetězce vyprázdní a cyklus se ukončí a pokračuje běh hlavního programu, kdy je změněna hodnota proměnné mám vysílat (mamVysilat) podle výsledku tohoto cyklu.

PWM je zde řešeno v novém vlákně, kdy je nadefinováno vlákno nazvané PWMvlakno, ve kterém je realizovaná PWM.

```
//nove vlakno PWM
          Thread PWMvlakno = new Thread(PWMcyklus);
          //spusteni PWM
          PWMvlakno.Start();
```

Při spuštění PWM vlákna se inicializuje délka periody a pak se nadefinuje proměnná sloužící pro výpočet zapnutého cyklu PWM.

```
//delka periody PWM v milisekundach
          const int delkaPeriody = 15000;
          //zasobnik pro PWM
          float pwmBuffer;
```

Deklarace výstupních pinů, které se používají na PWM. Pin led slouží pro indikaci, respektive kontrolu fungujícího PWM. Druhá deklarace pinu slouží jako výstupní port, který se zapíná a vypíná a tím realizuje PWM cyklus.

```
//deklarace diody
  OutputPort led = new OutputPort(Pins.ONBOARD_LED, false);
  //deklarace spinaciho pinu
  OutputPort spinac = new OutputPort(Pins.GPIO_PIN_D5, false);
```

Nastavení PWM je omezeno jen na hodnoty 0 - 1, přičemž 1 je 100 %. Pro střídu se používá pojem duty cyklus, který označuje dobu po jakou je zapnuto v procentech, ale to záleží na programové realizaci PWM. PWM lze naprogramovat pomocí nastavení doby zapnuto v procentech nebo nadefinováním délky dob zapnuto a vypnuto.

Příklad kódu ukazuje řešení zadání délky doby zapnutého a vypnutého pinu.

```
  while(true)
       {
           //aktivace pinu pro PWM
           spinac.Write(true);
           Thread.Sleep((int) (pwmBuffer * delkaPeriody));
           //vypnuti PWM
```

```
            spinac.Write(false);
            //nastaveni delky doby vypnute pwm
            Thread.Sleep((int)(delkaPeriody - pwmBuffer * delkaPeriody));
    }
```

PWM lze řešit i pomocí jiné varianty, kdy se použije příkaz:

```
PWM priklad = new PWM(Pins.GPIO_PIN_D5, délka periody, délka trvání,
false);
```

nebo

```
PWM priklad = new PWM(Pins.GPIO_PIN_D5, frekvence, střída, false);
```

Celý PWM cyklus je vložen v samostatném programovém vláknu. Důvod jeho použití je ve funkci hlavního programu, kdy je na konci smyčky While vložen sleep (pauza). Hlavní smyčka programu je využívána pro odesílání naměřené teploty do počítače každou jednu sekundu. Vložením PWM cyklu do hlavní smyčky programu, která odesílá naměřenou teplotu, by došlo k prodloužení tohoto cyklu o délku periody. Poté by program odesílání teploty do počítače nefungoval správně, neodesílal by naměřená data každou jednu sekundu. Řešením tohoto problému je právě nové vlákno. Nastavení konstant pro PI regulátor je v následující části kódu. Konstanty jsou dosazeny ze simulace v Matlab - Simulink.

```
//PI regulator
        //pid regulator, dosazene vypocitane parametry slozek P a I
        const double Kp = 2.6;
        const double Ki = 0.0021;
```

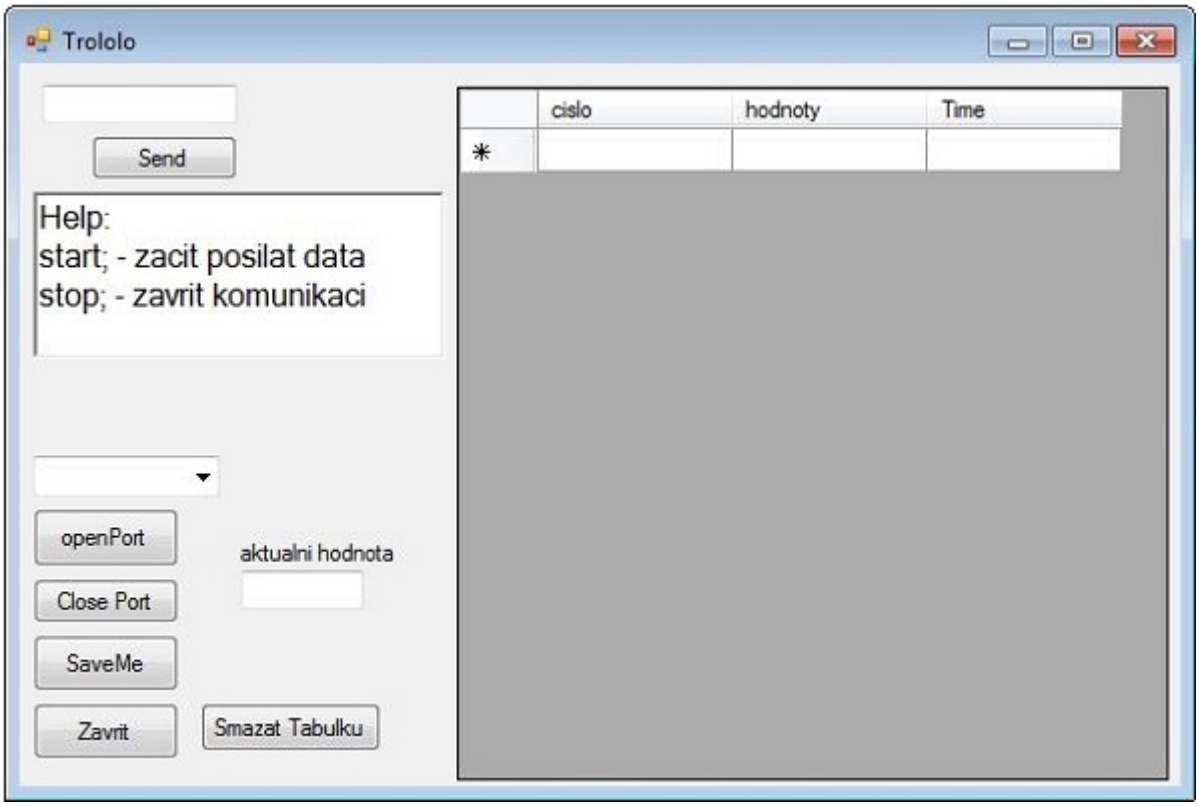
Cyklus PWM je řízen následujícím cyklem. Na jeho začátku se vypočítá rozdíl zadané a aktuální teploty. Poté se podle výsledku rozdílu teplot přepočítá složka P (hodnotaKp) a složka I (hodnotaKi).

```
        rozdilTeplot = zadanaTeplota - aktualniTeplota;
        //vypocet hodnot slozek
        double hodnotaKp = rozdilTeplot * Kp;
        hodnotaKi += rozdilTeplot * Ki;
        //vypocet PWM
        double PWM = hodnotaKp + hodnotaKi;
```

Na konci tohoto cyklu se PWM změní podle součtu přepočítaných složek hodnotaKp a hodnotaKi. Hodnota PWM se použije k přenastavní délky PWM cyklu.

5.0.2. Aplikace pro PC

Na následujícím obrázku (obr.5.1) je zobrazena vytvořená aplikace pro počítač. Celý kód je vložen jako příloha B.



Obr. 5.2.: Aplikace pro počítač.

Po připojení netduina a zapnutí aplikace se musí vybrat nejprve port, na kterém je netduino připojeno, respektive zařízení Ftdi, které vytváří virtuální sériový port, pak se použije tlačítko OpenPort na otevření portu ke komunikaci.

Tato aplikace se používá na odeslání příkazu netduinu tlačítkem send, podle konkrétního příkazu vykoná danou činnost. Nyní je naprogramováno jen zapnutí a vypnutí celého procesu. Při vysílání aplikace přijímá naměřené hodnoty z netduina a zapisuje je do tabulky. Celá tabulka může být exportována do souboru ve formátu pro excel tlačítkem SaveMe.

Použité knihovny při psaní aplikace pro windows.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
```

```
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
using System.Windows.Forms.DataVisualization.Charting;
using System.Data.SqlClient;
using Excel = Microsoft.Office.Interop.Excel;
```

V porovnání s knihovnami pro netduino je zde především rozdíl v knihovnách, které se používají na zobrazení výsledků. Knihovna Windows Forms umožňuje vytvoření rozhraní a knihovna System Drawing umožňuje vykreslení základních tvarů. Netduino takové knihovny nepoužívá, protože nemá operační systém windows a ani žádný displej. Oba programy, pro počítač i pro netduino, používají knihovnu System IO ports, která umožní používat například sériový port. Knihovna Excel je zde, aby bylo možné výsledky z aplikace v počítači exportovat do programu Excel.

Následující kód slouží k odesílání příkazů do netduina při zmáčknutí tlačítka Send. Nejprve se zapíše hodnota v texboxu, ke které je přidán středník. Středník slouží k rozpoznání, zda jde o příkaz. Po odeslání příkazu se zapne časovač, slouží k zápisu hodnot do tabulky. Poté se ještě vynuluje počítadlo (pocitadloInit), které je použité na mazání počátečních chybných hodnot. Počítadlo špatných hodnot je nutné, protože na začátku vysílání se inicializuje měření a přichází nesprávná data. Tento problém se pak řeší jejich smazáním.

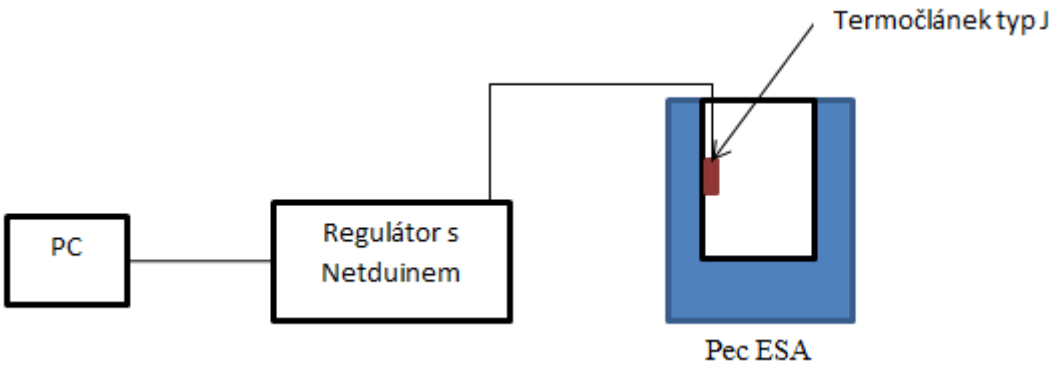
```
//Poslat znak primo k zadanemu textu v textBoxu.
      portik.Write(sendbox.Text + ";");
      //zapnuti casovace
      timeforeload.Enabled = true;
      //inicializace pocitadla spatnych hodnot
      pocitadloInit = 0;
```

6. Porovnání modelu z Matlabu a regulované soustavy

Tato kapitola obsahuje popis měření při ověřování naprogramovaného regulátoru a prezentaci naměřených hodnot. Poté následuje porovnání naměřených hodnot s hodnotami ze simulace.

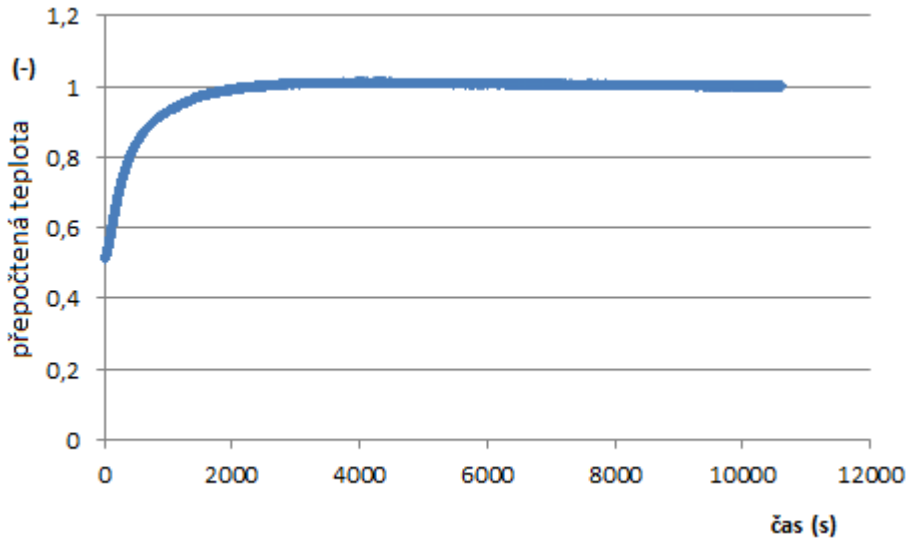
6.1. Ověření funkčnosti regulátoru na tepelné soustavě

Pro ověření naprogramovaného regulátoru byla použita pec ESA. Schéma zjednodušeného zapojení při ověřování funkčnosti regulátoru je vyobrazeno na obr.6.1. Ve schématu má Netduino připojeno termočlánek na snímání teploty a je napojeno na SSR relé, které ovládá spínání pece pomocí naprogramovaného regulátoru. Regulátor je řízen PI regulátorem s PWM. Naměřená data se posílají z Netduina do připojeného počítače. V počítači běží aplikace pro spuštění regulace a zároveň se v ní vypisují naměřená data.



Obr. 6.1.: Zjednodušené schéma zapojení při ověřování funkčnosti regulátoru.

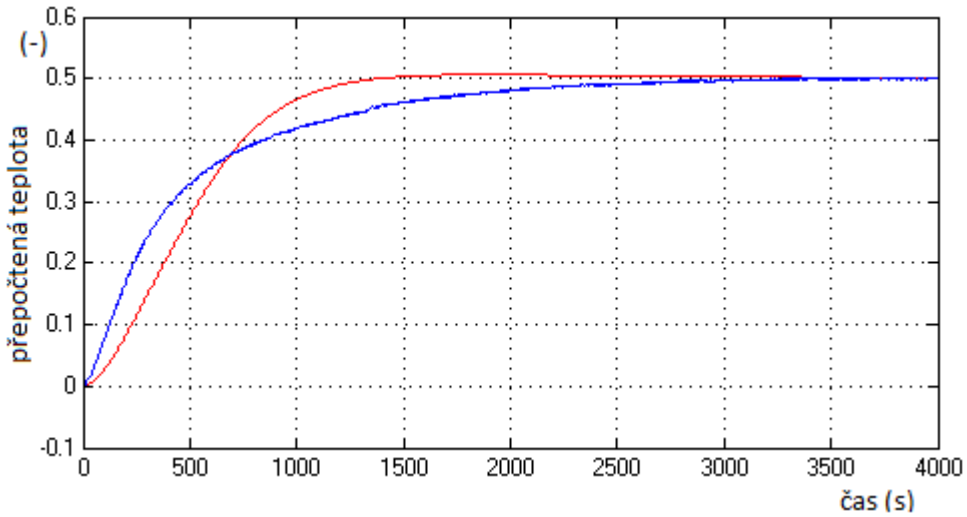
Graf naměřených hodnot z ověření funkčnosti regulátoru je v následujícím obr.6.2. V tomto grafu jsou hodnoty převedeny do jednotkového měřítka, kdy 1 znamená 400°C. Měření začalo v hodnotě 0,5, to je 200°C a bylo ukončeno po 176 minutách, tomu odpovídá hodnota v grafu 10560 sekund. Počáteční teplota pece při začátku měření byla 205°C, protože se pec zcela nevychladila od předchozího měření.



Obr. 6.2.: Naměřené hodnoty při ověřování fuknčnosti regulátoru.

6.2. Porovnání výsledků

Porovnání výsledků z ověřování fukčnosti regulátoru a výsledku ze simulace v Matlabu se nachází na (obr.6.3).

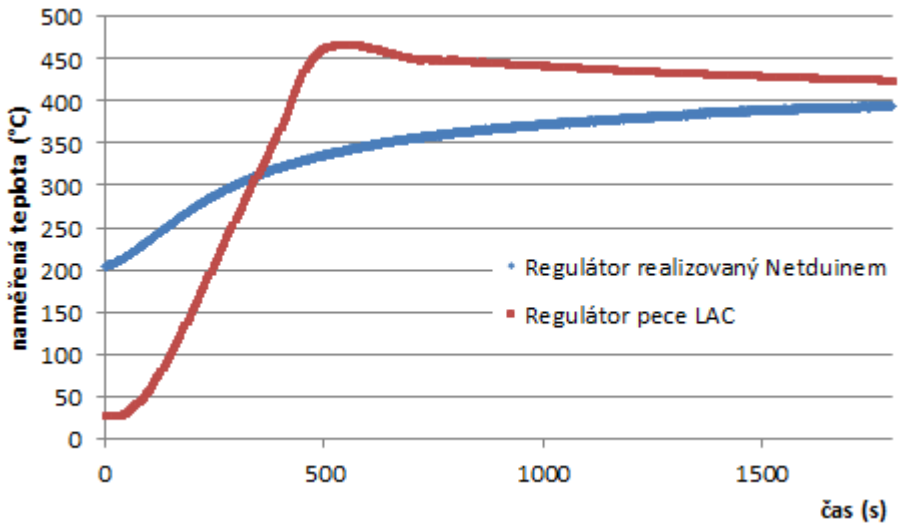


Obr. 6.3.: Graf porovnání naměřených hodnot a hodnot ze simulace (Červená křivka jsou hodnoty ze simulace, modrá křivka jsou naměřené hodnoty při ověřování regulátoru).

Pro srovnání byly hodnoty z ověřovacího měření přepočítány do stejného měřítka jako jsou hodnoty z Matlabu. Výsledek toho přepočtu byl vložen do Matlabu a vykreslen v jednom grafu spolu s hodnotami ze simulace. Následně je zde porovnání hodnot z měření pece LAC s regulátorem porovnáno s realizovaným regulátorem na peci ESA.

Z výsledku tohoto porovnání v obr.6.3 je patrné, že obě křivky nejsou zcela stejné. Za rozdíl mezi křivkami můžou především počáteční podmínky při ověřovacím měření funkčnosti regulátoru. Pec nebyla před začátkem měření zcela vychladlá, tj. teplota vnitřního prostoru pece nebyla menší než 30°C.

Křivky je nicméně možné srovnat alespoň podle překmitu, který je na grafu v obr.6.3 vidět. Hodnoty překmitu jsou u ověřovacího měření $4,5$°C, u hodnot ze simulace činí překmit 5°C.



Obr. 6.4.: Porovnání realizovaného regulátoru na peci ESA a regulátoru na peci LAC.

Je možné srovnat realizovaný regulátor, v grafu regulátor realizovaný netduinem, s křivkou ohřevu z pece LAC, která má vlastní regulátor Ht40 AL. Toto srovnání je na obr.6.4.

Přesto, že jsou křivky vzhledem k počátečním podmínkám rozdílné, je vidět jejich maximální překmit. Pro pec LAC je hodnota překmitu, měřená termočlánkem typu J, 68°C, zatímco pec ESA s realizovaným regulátorem má překmit pouhých $4,5$°C. V tomto srovnání je regulátor na peci ESA lepší.

Tvar obou křivek je dán typem použitého regulátoru. Regulátor realizovaný na peci ESA je PI regulátor a pec LAC měla regulátor PID.

7. Závěr

V této práci se povedlo vytvořit simulaci regulátoru v programu Matlab. Poté byl realizován regulátor pro tepelnou soustavu na platformě netduino a následně ověřena jeho funkčnost. Ověření funkčnosti regulátoru probíhalo na zlatnické peci (pec ESA).

Při ověřování funkčnosti realizovaného regulátoru dokázal naprogramovaný regulátor úspěšně regulovat teplotu uvnitř pece na zadanou teplotu. Během ověřování vytvořená aplikace pro počítač zaznamenávala naměřená data ohřevu pece. Naměřená data byla následně použita pro srovnání s hodnotami ze simulace v Matlabu. Poté byla naměřená data porovnána s výsledkem měření na peci LAC.

Funkčnost realizováného regulátoru byla ověřena pro cílovou teplotu 400°C. Z časových důvodů nešlo udělat více měření. Navíc data získaná při tomto měření jsou ovlivněna počátečními podmínkami.

Protože pec nestihla vychladnout od předchozího měření, počáteční teplota při regulaci činila 205°C. Pec tedy byla před měřením vyhřátá a tím byla ovlivněna celá charakteristika ohřevu. Z tohoto důvodu je porovnání dat ze simulace a dat z ověření funkčnosti realizovaného regulátoru omezeno jen na hodnotu překmitu. Překmit byl výrazně menší při srovnání realizovaného regulátoru s naměřenými hodnotami na peci LAC. Pro srovnání bylo použito měření na peci LAC s cílovou teplotou 400°C a při srovnání se simulací byl překmit téměř stejný.

Během ověřování realizovaného regulátoru byla na počítači spuštěna vytvořená aplikace pro ovládání netduina a tím byla vyzkoušena funkčnost tohoto programu. Aplikace dokázala úspěšně začít proces regulace a následně ho i ukončit. Během celého procesu regulace zaznamenávala naměřená data do tabulky a následně bylo tyto data možné vyexportovat do excelu.

Pro použití naprogramované aplikace na ovládání netduina by bylo dobré doplnit funkci zadávání cílové teploty, aby nebylo potřeba ji zadávat napevno do programu netduina.

Program pro netduino fungoval správně. Drobné úpravy by mohl doznat snad jen model tepelné soustavy, který není zcela správný, neboť byl vytvořen z neúplné přechodové charakteristiky. Přechodovou charakteristiku totiž nebylo možné naměřit až do jejího ustálení vzhledem k obavám z možného poškození pece.

I přes tyto drobné nedostatky se podařilo sestrojit ověřitelně funkční regulátor.

Literatura

[1] KREIDL, K. *Měření teploty*. 1. vyd. Praha: BEN - technická literatura, 2005. 240 s. ISBN 80-7300-145-4.

[2] FUKÁTKO, T, FUKÁTKO, J. *Teplo a chlazení v elektronice II*. Přel. F. Štráfeflda. 1. vyd. Praha: BEN - technická literatura, 2006. 120 s. ISBN 80-7300-199-3.

[3] Magazín 3 PÓL: *Elektřina přímo z tepla* [online]. Magazín 3 PÓL: ©2010 [cit. 27.2.2014]. Dostupné z: `http://3pol.cz/888-elektrina-primo-z-tepla`

[4] Franklin, G, Powell, D, Emami-Naeini, A *Feedback Control of Dynamic Systems*. 6. vyd. Michiganská univerzita: Prentice Hall, 2009. 840 s. ISBN 978-0136019695.

[5] KREIDL, M, ŠVARC, R. *Technická diagnostika*. 1. vyd. Praha: BEN - technická literatura, 2006. 408 s. ISBN 80-7300-158-6.

[6] TŮMOVÁ, O. *Metrologie a hodnocení procesů*. 1. vyd. Praha: BEN - technická literatura, 2009. 232 s. ISBN 978-80-7300-249-7.

[7] VALTER, J. *Regulace v praxi aneb Jak to dělám já*. 1. vyd. Praha: BEN - technická literatura, 2010. 176 s. ISBN 978-80-7300-256-5.

[8] BURKHARD, K. *Měření, řízení a regulace pomocí PC*. Přel. V. Losík. 1. vyd. Praha: BEN - technická literatura, 2005. 272 s. ISBN 80-7300-089-X.

[9] ŠVARC, T. *Základy automatizace*. Učební texty pro kombinovanou formu bakalářského studia. Brno: VUT - Fakulta strojního inženýrství, 2002. 102 s.

[10] Distefano, J, Stubberud, R, Williams, I *Theory and problems of Feedback and control systems*. 2. vyd. Kalifornská univerzita: McGraw-Hill, 1995. 512 s. ISBN 0-07-017052-5.

[11] HABERLE, H, a kol. *Průmyslová elektronika a informační technologie*. 1. vyd. Praha: Europa - Sobotáles cz. s.r.o., 2003. 720 s. ISBN 80-86706-04-4.

[12] Digitální knihovna VUT v Brně: *Kinetika krystalizace kopolymerů* [online]. Digitální knihovna VUT v Brně: ©2011 [cit. 22.2.2014]. Dostupné z: https://dspace.vutbr.cz/xmlui/bitstream/handle/11012/2074/Kinetika%20krystalizace%20kopolymer%C5%AF%20PP_Krajcik.pdf?sequence=1

[13] Knihovna UTB ve Zlíně: *Stereologie nadmolekulární struktury polymerů* [online]. Knihovna UTB ve Zlíně: ©2011 [cit. 25.2.2014]. Dostupné z: https://dspace.k.utb.cz/bitstream/handle/10563/4325/matou%C5%A1ek_2007_bp.pdf?sequence=1

[14] INTECH - Open Access Company: *Johnson-Mehl-Avrami plot of the ball-milled* [online]. INTECH: ©2014 [cit. 25.2.2014]. Dostupné z: http://www.intechopen.com/books/applications-of-calorimetry-in-a-wide

[15] GM Electronic: *KSD240AC8* [online]. GME: ©2014 [cit. 15.3.2014]. Dostupné z: http://www.gme.cz/ssr-pro-montaz-na-chladic/ksd240ac8-p635-017/#popis

[16] PV Electronic: *MOD - TC* [online]. PV Electronic: ©2014 [cit. 15.3.2014]. Dostupné z: http://www.pvelectronic.eu/Podle-Vyrobcu/OLIMEX/UEXT-Modules-1/MOD-TC.html?listtype=search&searchparam=mod-tc

[17] Netduino: *Netduino Plus 2* [online]. Netduino: ©2014 [cit. 15.3.2014]. Dostupné z: http://netduino.com/netduinoplus2/specs.htm

[18] FITkit: *Firmware / Komunikační systém* [online]. FITkit: ©2014 [cit. 29.3.2014]. Dostupné z: http://merlin.fit.vutbr.cz/FITkit/docs/firmware/fpga_interconect.html

[19] Sparkfun: *FTDI Basic Breakout* [online]. Sparkfun: ©2014 [cit. 15.3.2014]. Dostupné z: https://www.sparkfun.com/products/9873

[20] MathWorks: *MatLab* [online]. MathWorks: ©2014 [cit. 16.3.2014]. Dostupné z: http://www.mathworks.com/help/matlab/index.html?s_cid=BB

[21] DHservis: *Pulzně šířková modulace* [online]. DHservis: ©2014 [cit. 16.3.2014]. Dostupné z: http://www.dhservis.cz/psm.htm

[22] Mik, P: *Návrh adaptivních PID regulátorů* [online]. ČVUT v Praze: ©2010 [cit. 18.3.2014]. Dostupné z: http://support.dce.felk.cvut.cz/mediawiki/images/c/c3/Dp_2010_mik_petr.pdf

[23] Stackoverflow: *Excel export* [online]. Stackoverflow: ©2014 [cit. 25.3.2014]. Dostupné na: `http://stackoverflow.com`

[24] Mahrlo: *Ht40AL Programový PID regulátor* [online]. Mahrlo: ©2014 [cit. 28.3.2014]. Dostupné z: `http://marweb.sk/regulatory/702-ht40al-programovy-pid-regulator.html`

Seznam zkratek

u	akční veličina
w	požadovaná veličina
e	regulační odchylka
y	výstupní (regulovaná) veličina
v	poruchová veličina
PWM	pulzně šířková modulace (pulse width modulation)
AD	analogově digitální převodník
JMAK	Johnson Mehl Avrami Kolmogorov, označení Avramiho rovnice
SSR	solid state relay
AC	střídavý proud (alternating current)
DC	stejnosměrný proud (direct current)
SPI	sériové periferní rozhraní(Serial Peripheral Interface)
EEPROM	elektricky mazatelná nevolativní paměť (Electrically Erasable Programmable Read-Only Memory)
MOSI	data vystupující z master zařízení (Master Out, Slave In)
MISO	data vstupující do master zařízení (Master In, Slave Out)
SCLK	hodinový signál, určuje podle režimu, kdy dojde k vzorkování dat
avr	aproximovaná data Avramiho rovnicí
Netduino (Netduino Plus 2)	open source elektronická platforma fungující pomocí NET Micro Framework. Deska je vybavena 32-bitovým mikrokontrolérem s vývojovým prostředím.

Seznam obrázků

2.1. Schéma Seebeckova jevu. 7
2.2. Schéma regulačního obvodu. 11
2.3. Demonstrace pojmů u PWM při nastavení střídy na 50%. 12
2.4. Chování PWM. 13
2.5. Přechodová charakteristika P a PI regulátoru. 14
2.6. Přechodová charakteristika PID regulátoru. 15
2.7. Avramiho rovnice - transformace exponenciály na přímku. 18
2.8. Návod na Strejcovu metodu. 18
2.9. Úvodní obrazovka při spuštění nadstavby Matlab - Simulink. 20
2.10. Návrh regulátoru v pidtool. 21

3.1. Pec LAC s Ht40 AL při měření. 23
3.2. Regulátor Ht40 AL. 25
3.3. SSR relé. 26
3.4. MOD - TC. 27
3.5. Netduino Plus 2. 28
3.6. FTDI Basic Breakout. 28
3.7. Fotka zapojení z měření na peci LAC. 30
3.8. Schéma pracovního prostoru pece. 31
3.9. Měření na peci ESA. 31
3.10. Schéma zapojení měření. 32
3.11. Umístění termočlánků v pracovním prostoru pece. 33

4.1. Příklad výsledku avr aproximace s měřenou teplotou na 700°C 35
4.2. Výsledek porovnání naměřených a aproximovaných hodnot pro 700°C. . . 36
4.3. Naměřené hodnoty na peci ESA termočlánky J a K. 37
4.4. Ukázka simulace odezvy na jednotkový skok v Matlabu. 37
4.5. Výsledek odezvy na jednotkový skok pro 800°C. 38
4.6. PI regulátor namodelovaný v Matlab - Simulink. 39
4.7. Vypočítané parametry od PIDtool. 40
4.8. Výsledek regulace s dodatečnými parametry. 41

4.9. PI regulátor bez PID bloku v prostředí Matlab - Simulink. 41
4.10. Výsledek regulace s ručně měněnými parametry P a I. 42

5.1. Zjednodušené schéma zapojení řídícího obvodu. 43
5.2. Aplikace pro počítač. 49

6.1. Zjednodušené schéma zapojení při ověřování funkčnosti regulátoru. . . . 51
6.2. Naměřené hodnoty při ověřování fuknčnosti regulátoru. 52
6.3. Graf porovnání naměřených hodnot a hodnot ze simulace 53
6.4. Porovnání realizovaného regulátoru na peci ESA s pecí LAC. 54

Seznam tabulek

2.1. Rozsahy měřených teplot u vybraných termočlánků. 9
2.2. Tabulka konstant pro Strejcovu metodu. 19
2.3. Tabulka pro určení konstanty τ_2. 19

3.1. Štítkové údaje pece LAC. 22
3.2. Štítkové údaje pece ESA. 23
3.3. Přiřazení termočlánků na peci LAC. 25
3.4. Přiřazení termočlánků na peci ESA. 25
3.5. Použité termočlánky. 25

4.1. Údaje k naměřeným a aproximovaným křivkám. 34
4.2. Hodnoty u termočlánků typu J. 35
4.3. Tabulka vypočítaných konstant pro přenos. 39

A. Kód pro netduino

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using SecretLabs.NETMF.Hardware;
using SecretLabs.NETMF.Hardware.Netduino;
using System.Text;
using System.IO;
using System.IO.Ports;

namespace NetduinoRegulace
{
public class Program
    {
        //nastaveni PWM, strida
        //0.0 = 0\%, 1.0 = 100\%
        static float nastavenaPWM = 0.0f;
        //pozadovana teplota ve stupnich celsia, 0 znamena vypnuto
        static int zadanaTeplota=100;
        static bool mamVysilat = true;
        //definice serioveho portu
        static SerialPort serial = new SerialPort
        (SerialPorts.COM1, 9600, Parity.None, 8, StopBits.Two);
        //promenna pro cteni prichoziho retezce
        static string lineFin = "";
        //promenna pri PID regulator
        static double hodnotaKi = 0;

        //=====================================
```

```
/// <summary>
/// Hlavni kod programu ridici Netduino
/// </summary>
//=====================================

public static void Main()
{

    //nastaveni portu SPI pro pripojeni termoclanku
    \#region SPI Definice
    SPI.Configuration Device1 = new SPI.Configuration(
    Pins.GPIO_PIN_D10, // vybrany pin
    false,             // aktivni stav v nule
    0,                 // nastaveni casu portu
    0,                 // doba setrvani na portu
    true,              // klidovy stav hodin
    true,              // vzorkovaci cas okraje
    250,               // taktovací frekvence v KHz
    SPI_Devices.SPI1   // pouzita sbernice
    );
    \#endregion

    //deklarace promenne pouzivajici SPI
    SPI SPIBus = new SPI(Device1);
    SPIBus.Config = Device1;

    //nove vlakno PWM
    Thread PWMvlakno = new Thread(PWMcyklus);
    //spusteni PWM
    PWMvlakno.Start();

    double aktualniTeplota;

    //spusteni preruseni na kontrolu prichoziho retezce
    serial.DataReceived += new SerialDataReceivedEventHandler
    (serial_DataReceived);
    //otevreni serioveho portu
```

```
serial.Open();


while(true)
{

    //definice bufferu
    byte[] WriteBuffer = new byte[2];
    byte[] ReadBuffer = new byte[2];

    //odeslani namerene hodnoty termoclankem do aplikace v PC
   \#region Odesilani a konvertor dat Teploty

    SPIBus.WriteRead(WriteBuffer, ReadBuffer);
    int data;
    //rotace bitu a nasledna uprava vysledne hodnoty
    data = (short)(ReadBuffer[0] << 5 | ReadBuffer[1] >> 3);
    //tempC = data / 4.0; //prevod honoty na spravne cislo;
    aktualniTeplota = data;
    //debug zobrazeni hodnot
    //Debug.Print(tempC.ToString());
    byte[] TxB;
    //TxB = System.Text.Encoding.UTF8.GetBytes
    (tempC.ToString() + "\r\n");

    //odeslani hodnoty do do pocitace
    TxB =System.Text.Encoding.UTF8.GetBytes
    (aktualniTeplota.ToString());
    if (mamVysilat == true)
    {
        serial.Write(TxB, 0, TxB.Length);
    }
    Debug.Print("aktualni teplota: " + aktualniTeplota/4);

    \#endregion

    //PI regulator
```

```
                //pid regulator, dosazene vypocitane parametry
                const double Kp = 2.6;
                const double Ki = 0.0021;

                //prepocet hodnoty na stupne
                aktualniTeplota = aktualniTeplota / 4;

                //spusteni regulace
                //zadana teplota musi byt vetsi nule, zaporna byt nemuze
                if (zadanaTeplota > 0)
                {
                    double rozdilTeplot;
                    rozdilTeplot = zadanaTeplota - aktualniTeplota;
                    //deleno tisicem vzhledem k prepoctum
                    //do meritka 0 - 1 v matlabu pri vypoctech
                    rozdilTeplot = rozdilTeplot / 1000;

                    //vypocet hodnot slozek Kp a Ki
                    double hodnotaKp = rozdilTeplot * Kp;

                    hodnotaKi += rozdilTeplot * Ki;

//prepocet delky PWM
                    double PWM = hodnotaKp + hodnotaKi;

                    //osetreni maximalni a minimalni hodnoty pro PWM
                    //cyklusnejde zadat vice nez 1
                    if (PWM > 1)
                    {
                        PWM = 1;
                    }

                    //nelze mit zaporne nastaveni pulzu PWM
                    else if (PWM < 0)
                    {
                        PWM = 0;
                    }
```

```
//prenastaveni PWM periody
                    nastavenaPWM = (float)PWM;

                }

                //osetreni v pripade zadani hodnoty mensi 0
                else
                {
                    nastavenaPWM = 0f;
                }

                Thread.Sleep(1000);

            }

        }

        //======================================
        /// <summary>
        /// Vlakno realizujici PWM
        /// </summary>
        //======================================

    static void PWMcyklus()
    {
        //delka periody PWM v milisekundach
        const int delkaPeriody = 15000;
        //zasobnik pro PWM
        float pwmBuffer;
        //definice diody
        OutputPort led = new OutputPort(Pins.ONBOARD_LED, false);
        //definice spinaciho pinu
        OutputPort spinac = new OutputPort(Pins.GPIO_PIN_D5, false);

        while (true)
        {
            pwmBuffer = nastavenaPWM;
            //zapnuti diody
```

```
            led.Write(true);
            //aktivace pinu pro PWM
            spinac.Write(true);
            Thread.Sleep((int) (pwmBuffer * delkaPeriody));
            //vypnuti diody a pinu pro PWM
            led.Write(false);
            spinac.Write(false);
            //nastaveni delky doby vypnute pwm
            Thread.Sleep((int)(delkaPeriody - pwmBuffer * delkaPeriody));

        }

    }

        static void serial_DataReceived
        (object sender, SerialDataReceivedEventArgs e)
{
        char cteciByte;

        while(serial.BytesToRead > 0)
        {

            //cteni prichozich znaku ze serioveho portu
            cteciByte = (char) serial.ReadByte();

            //cte se seriovy port dokud neprijde znak nesplnujici
            //podminku
            if (cteciByte != ';')
            {
                lineFin += cteciByte;
            }

            //pokud znak splnujici podminku,
            //tak se kontroluje prichozi retezec pro start programu
            else
            {
                if (lineFin.CompareTo("start") == 0)
                {
```

```
                    mamVysilat = true;
                    Debug.Print("start programu");
                }
                else if (lineFin.CompareTo("stop") == 0)
                {
                    mamVysilat = false;
                    zadanaTeplota = 0;
                    Debug.Print("konec programu");
                }

                //cisteni promenne po kazdem pruchodu cyklu
                lineFin = "";

            }
        }
    }
  }
}
```

B. Kód aplikace pro počítač

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
using Excel = Microsoft.Office.Interop.Excel;

namespace comboboxserialports
{
    public partial class Form1 : Form
    {
        //deklarace promennych
        SerialPort portik;
        public int pocitadloInit = 0;
        public double predchoziH = 1;

        public Form1()
        {

            InitializeComponent();
            //definice serioveho portu
            portik = new SerialPort();
            //nastaveni casovace na vysilani hodnot
            timeforeload.Interval = 1000;

        }
```

```
private void nacteniformulare(object sender, EventArgs e)
{
    string[] myserial;
    //nacteni dostupnych seriovych portu
    myserial = System.IO.Ports.SerialPort.GetPortNames();
    vyberPortu.Items.AddRange(myserial);
    //nastaveni defaultni hodnoty, aby byl vzdy vybran port
    vyberPortu.SelectedIndex = vyberPortu.Items.Count - 1;

}

private void potvrdvyber(object sender, EventArgs e)
{
    //nastaveni seroveho portu
    #region Serial Port nastaveni
    portik.BaudRate = 9600;
    portik.Parity = Parity.None;
    portik.StopBits = StopBits.Two;
    portik.ReadTimeout = 1000;
    #endregion

    //vyjimka pri spousteni spustene komunikace
    try
    {
        portik.PortName = vyberPortu.SelectedItem.ToString();
        portik.PortName.ToString();
        portik.Open();
    }
    //pokud je port otevren
    catch
    {
        portik.Close();
        portik.Open();
    }
}

private void poslani(object sender, EventArgs e)
```

```
{

    //Poslat znak primo k zadanemu textu v textBoxu.
    portik.Write(sendbox.Text + ";");
    //zapnuti casovace
    timeforeload.Enabled = true;
    //inicializace pocitadla spatnych hodnot
    pocitadloInit = 0;

}

private void CloseButton_Click(object sender, EventArgs e)
{
    //uzvreni portu, aby aplikace nepadla pokud by se zaviral
    //neotevreny port
    if (portik.IsOpen)
    {
        //uzavreni serioveho portu
        portik.Close();
        //vypnuti casovace
        timeforeload.Enabled = false;
        MessageBox.Show("JE Zavreno");
    }
    //pokud neni seriovy port otevren
    else
    {
        MessageBox.Show("Neni co zavirat");
    }
}

private void ZavriApp_Click(object sender, EventArgs e)
{
    //zavreni aplikace
    Close();
    //zavreni portu
    portik.Close();
    //ukonceni casovace
```

```
    timeforeload.Enabled = false;
}

/// <summary>
/// Vypis hodnot.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void timer1_Tick(object sender, EventArgs e)
{
    //Nacteni hodnot do zasobniku (bufferu) a prevedeni
    //hodnot na string
    #region Nacteni hodnot do Bufferu a prevedeni na String.
    string lineFin = "";
    int bytesToRead = portik.BytesToRead;
    byte[] buffer = new byte[bytesToRead];
    portik.Read(buffer, 0, buffer.Length);
    //nacitani hodnot a jejich prevedeni na string
    lineFin += "" + new string(Encoding.UTF8.GetChars(buffer));
    #endregion

    //promenna pro prepocet cisla na spravnou teplotu
    double tempC;

    //prepocet na stupne
    try
    {
        //pretypovani hondoty
        tempC = Convert.ToDouble(lineFin);
        tempC = tempC / 4;
    }
    //vyjimka pokud prijde spatne cislo
    catch
    {
        tempC = 000;
    }

    //cislovani radek v tabulce
```

```
#region Cisla radku v tabulce.
//cislovani od 1
int rowNumber = 1;
foreach (DataGridViewRow row in dTGHodnoty.Rows)
{
    if (row.IsNewRow) continue;
    row.HeaderCell.Value = rowNumber;
    rowNumber = rowNumber + 1;
}
//nastaveni automatickeho prizpusobeni velikosti sloupce
dTGHodnoty.AutoResizeRowHeadersWidth
(DataGridViewRowHeadersWidthSizeMode.AutoSizeToAllHeaders);
#endregion

//nastaveni formatu casu zapisovaneho do tabulky
#region Format casu v tabulce.
DateTime dT = DateTime.Now;
string tiktak = dT.ToString("HH:mm:ss");
#endregion

//Oprava prichozich hodnot.
if (tempC > 1000)
{
    //nahrazeni aktualni hodnoty predchozi hodnotou
    tempC = predchoziH ;
}
//prechozi hodnota ulozena do promenne
predchoziH = tempC;

//pretypoveni hodnoty, do tabulky lze zapisovat jen string
tbPredchoziH.Text = predchoziH.ToString();

//vypsani dat do tabulky.
dTGHodnoty.Rows.Add(rowNumber, tempC , tiktak);

//pocitadlo na odstraneni prvnich 5 chybnych hodnot
if (pocitadloInit < 5)
```

```
        {
            pocitadloInit++;
            dTGHodnoty.Rows.Clear();
        }
    }

    private void ClearRTBox_Click_1(object sender, EventArgs e)
    {
        //vymazani hodnot z tabulky
        dTGHodnoty.Rows.Clear();
    }
```

Export do Excelu dle [23]:

```
private void saveToExcel_Click(object sender, EventArgs e)
  {
      Excel.Application xlApp;
      Excel.Workbook xlWorkBook;
      Excel.Worksheet xlWorkSheet;
      object misValue = System.Reflection.Missing.Value;

      xlApp = new Excel.Application();
      xlWorkBook = xlApp.Workbooks.Add(misValue);
      xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);

      int i = 0;
      int j = 0;

      //zapisovani hodnot z tabulky do excelu
      for (i = 0; i <= dTGHodnoty.RowCount - 1; i++)
      {
          for (j = 0; j <= dTGHodnoty.ColumnCount - 1; j++)
          {
              DataGridViewCell cell = dTGHodnoty[j, i];
              xlWorkSheet.Cells[i + 1, j + 1] = cell.Value;
          }
      }

      xlWorkBook.Close(true, misValue, misValue);
```

```
        xlApp.Quit();

        releaseObject(xlWorkSheet);
        releaseObject(xlWorkBook);
        releaseObject(xlApp);
    }
```

Ošetření vyjímky pro export do Excelu dle [23]:

```
        private void releaseObject(object obj)
        {
            //vyjimka na vytvareni souboru pri ukladani dat do excelu
            try
            {
                System.Runtime.InteropServices.Marshal.
                ReleaseComObject(obj);
                obj = null;
            }
            catch (Exception ex)
            {
                obj = null;
                MessageBox.Show
                ("Exception Occured while releasing object "
                + ex.ToString());
            }
            finally
            {
                GC.Collect();
            }
        }
    }
}
```

C. Obsah přiloženého CD

K této práci je přiloženo CD, na kterém jsou uloženy zdrojové kódy a data z měření.